УДК 004
ББК 32.973
И 74

Авторы-составители: Т. А. Заяц, ст. преподаватель кафедры информационно-вычислительных систем Белорусского торгово-экономического университета потребительской кооперации;
В. М. Заяц, ведущий инженер-электроник ИВЦ СОАО «Гомелькабель»

Рецензенты: Г. И. Наринский, начальник информационно-вычислительного центра СОАО «Гомелькабель»;
Е. А. Левчук, канд. техн. наук, доцент Белорусского торгово-экономического университета потребительской кооперации

Рекомендован научно-методическим советом учреждения образования «Белорусский торгово-экономический университет потребительской кооперации». Протокол № 3 от 8 февраля 2011 г.

И 74 **Информационные** ресурсы. Технологии работы с клиент-серверными СУБД (на примере СУБД MySQL) : практикум к лабораторным занятиям для студентов заочной формы получения высшего образования специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. : Т. А. Заяц, В. М. Заяц. – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2013. –72 с.
ISBN 978-985-540-058-6

УДК 004
ББК 32.973

ISBN 978-985-540-058-6

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Информационные ресурсы представляют собой документы и отдельные массивы документов в информационных системах. Знание состояния информационных ресурсов, умение работать на информационном рынке и умение использовать полученную информацию в своей деятельности – обязательное требование, предъявляемое к специалисту, так как информация является основой для принятия решений во всех областях предпринимательской деятельности и при решении задач управления экономикой в государственных структурах.

Специальная дисциплина «Информационные ресурсы» ставит своей целью не только ознакомить студентов с мировыми и отечественными информационными ресурсами, но и сформировать практические навыки разработки информационных ресурсов в корпоративных информационных системах и их использования на высоком профессиональном уровне.

Практикум состоит из трех разделов.

В первом разделе рассматриваются возможности системы управления базами данных (СУБД) MySQL – одной из наиболее популярных СУБД клиент-серверного типа; даются основные характеристики СУБД MySQL; указываются программные продукты, входящие в состав дистрибутива MySQL.

Второй раздел посвящен изучению языка SQL-запросов к базам данных. В разделе приводятся важнейшие конструкции языка SQL, используемые для формирования структуры базы данных и манипулирования информацией, хранящейся в ней. Завершается раздел несколькими лабораторными работами по созданию баз данных и выполнению запросов к ним.

В третьем разделе приводятся теоретические сведения по использованию средств языка веб-программирования PHP для взаимодействия с сервером MySQL; рассматривается синтаксис основных функций языка PHP; приводятся листинги программ на PHP, содержащие примеры установки соединения с сервером баз данных, создания запросов к базе данных, обработки и отображения результатов запросов к серверу, закрытия соединения с сервером MySQL. В разделе имеются лабораторные работы, позволяющие проверить степень усвоения полученного материала.

1. ОСНОВНЫЕ СВЕДЕНИЯ О СУБД MySQL

1.1. Справочная информация по СУБД MySQL

Одной из наиболее популярных систем управления базами данных является СУБД MySQL – реляционная СУБД типа «клиент-сервер».

Создателем СУБД MySQL является Майкл Видениус (также известный как Монти) из компании TcX DataKonsultAB (Стокгольм, Швеция). Начиная с 1995 г. MySQL стала одной из самых распространенных СУБД в мире, что отчасти обусловлено ее скоростью, надежностью и гибкой лицензионной политикой. Компания TcX заявляет, что СУБД MySQL используется примерно на 500 тыс. серверов (по состоянию на 2006 г.).

1.2. Состав дистрибутива СУБД MySQL

Дистрибутив MySQL можно загрузить из Интернета по адресу *http://www.mysql.com/downloads/index.html* или *http://www.mysql.ru*. В состав дистрибутива MySQL входят следующие программные продукты:

- *SQL-сервер*, обеспечивающий доступ к базам данных (принимает запросы клиентов, поступающие по сети, и осуществляет доступ к содержимому базы данных для предоставления информации, которую запрашивают клиенты);
- *клиентская программа* доступа к серверу, которая осуществляет подключение к серверу и передает запросы на сервер (это интерактивная программа, позволяющая делать запросы и просматривать полученные результаты);
- *административные и сервисные программы*, помогающие работать с СУБД.

1.3. Основные преимущества СУБД MySQL

Среди преимуществ СУБД MySQL выделяют следующие:

- *Быстродействие*. СУБД MySQL является одной из самых быстрых баз данных из имеющихся на современном рынке.
- *Простота использования*. СУБД MySQL является высокопроизводительной и относительно простой в использовании, которую значительно проще инсталлировать и администрировать, чем многие большие системы.

• *Поддержка языка запросов*. MySQL «понимает» команды языка SQL – языка запросов, нашедшего применение во всех современных СУБД.

• *Возможности обработки*. Количество строк в таблицах может достигать 50 млн. Компания-разработчик утверждает, что использует MySQL с 1996 г. на сервере с более 40 базами данных, которые содержат 10 тыс. таблиц, из которых более 500 имеют более 7 млн строк.

• *Возможности доступа*. Сервер позволяет одновременно подключаться неограниченному количеству пользователей, одновременно работающих с базой данных. Доступ к серверу СУБД MySQL можно осуществить в интерактивном режиме.

• *Поддержка интерфейса ODBC*. Доступ к базам данных СУБД MySQL возможен с помощью приложений, поддерживающих обобщенный интерфейс ODBC (Open Data Bases Connectivity – открытое взаимодействие с базами данных), который может использоваться для одновременного соединения с различными СУБД.

• *Взаимодействие и безопасность*. СУБД MySQL предназначена для работы в сети и может быть доступна через Интернет. Таким образом, обращаться к базе данных и работать с данными можно из любой точки земного шара. Но при этом СУБД MySQL снабжена развитой системой защиты от несанкционированного доступа. Для обеспечения дополнительной защиты СУБД MySQL поддерживает криптируемые соединения с использованием протокола SSL.

• *Аппаратная совместимость*. СУБД MySQL отлично работает как под управлением различных версий UNIX, так и под управлением таких систем, как Windows и OS/2. СУБД MySQL может работать как на домашних ПК, так и на мощных серверах.

• *Малый размер*. СУБД MySQL имеет ограниченный размер по сравнению с огромным дисковым пространством, необходимым большинству коммерческих СУБД (сервер Apache – 25 Мбайт, СУБД MySQL – 120, PHP – 150 Мбайт).

1.4. Терминология СУБД

Система управления базами данных СУБД MySQL классифицируется как реляционная система управления базами данных, или RDBMS (Relational Database Management System). Эта аббревиатура разделяется на части следующим образом:

• слово «реляционная», или R (Relational), обозначает популярную

разновидность СУБД, которые поддерживают базы с реляционной моделью организации данных в них;

• база данных, или DB (Database), – это именованная совокупность информации из некоторой предметной области, структурированной определенным способом;

• система управления, или MS (Management System), является компьютерной программой, позволяющей пользователю создавать, поддерживать базы данных и управлять ими; вставлять, выбирать, модифицировать и удалять записи.

2. ОСНОВНЫЕ ПРИЕМЫ РАБОТЫ В СУБД MySQL

2.1. Команды работы с базой данных

Работа с базой данных предполагает несколько этапов:

1. Создание (или инициализация) базы данных.
2. Создание таблиц в базе данных.
3. Взаимодействие с таблицами посредством запросов (выполнение операций вставки, выборки, модификации или удаления данных).

2.1.1. Команда создания базы данных

Команда создания базы данных имеет следующий синтаксис:

```
CREATE DATABASE имя_базы_данных;
```

Здесь `имя_базы_данных` является именем создаваемой базы данных. Любая команда в СУБД MySQL заканчивается символом «точка с запятой» (`;`).

Примечание – *Имя* может состоять из набора любых алфавитно-цифровых символов латинского языка, а также символов нижнего подчеркивания и доллара (_ и $). Имена могут начинаться с любого допустимого символа, включая цифры. Нельзя использовать точку и не использовать разделители при написании имени пути операционной системы Windows (/ или \).

Например, чтобы создать новую базу данных `forum`, нужно набрать команду, приведенную в листинге 1.

Листинг 1. Создание базы данных

```
CREATE DATABASE forum /*создание базы данных forum*/;
```

2.1.2. Команда просмотра списка баз данных

Для того чтобы убедиться, что база данных `forum` успешно создана, можно выполнить команду `SHOW DATABASES` (листинг 2), которая покажет, какие базы данных существуют в системе.

Листинг 2. Просмотр списка созданных баз данных

```
SHOW DATABASES;
```

Как видим, среди различных баз данных присутствует и только что созданная база данных `forum` (рисунок 1).

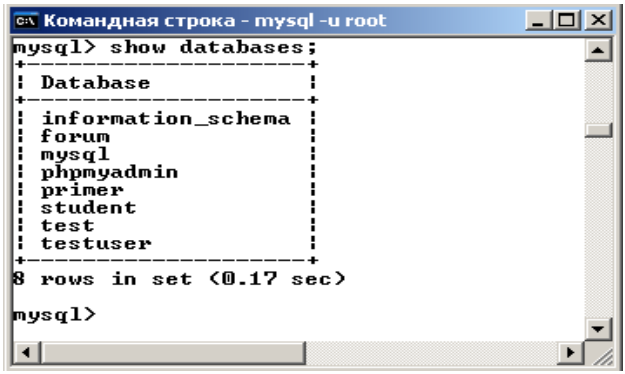

Рисунок 1 – **Базы данных, созданные в MySQL**

2.1.3. Команда активизации (выбора) базы данных

Для того чтобы начать работу с таблицами, необходимо сообщить MySQL, в какой базе данных пользователь намерен создавать свои таблицы. Это осуществляется при помощи команды `USE`:

```
USE имя_базы_данных;
```

Активизируем созданную базу `forum` (листинг 3).

Листинг 3. Выбор базы данных

```
USE forum;
```

Примечание – Студенты заочной формы получения высшего образования не имеют прав на создание новых баз данных. Администратором сети разрешено только использование уже имеющейся базы данных с именем `studentzaot`, расположенной на учебном сервере университета (uchserv). Для активизации указанной базы нужно воспользоваться следующей командой:

```
mysql> USE studentzaot;
```

2.2. Создание таблиц в базе данных

2.2.1. Команда создания таблиц в базе данных

Команда `CREATE TABLE` создает новую таблицу в выбранной базе данных. В общем случае команда имеет следующий синтаксис:

```
CREATE TABLE имя_таблицы (описание_столбца, … ,
описание_столбца) [дополнительные_опции_таблицы];
```

Здесь `имя_таблицы` – имя создаваемой таблицы.

Примечание – В квадратных скобках указываются необязательные параметры команды.

Синтаксис параметра `описание_столбца` в команде `CREATE TABLE` имеет следующий вид:

```
Имя_столбца тип_данных_столбца [параметры]
```

`Имя_столбца` всегда располагается в начале. Имя может содержать до 64 символов. Оно может состоять из совокупности алфавитно-цифровых символов в кодировке, принятой по умолчанию на сервере, символов подчеркивания и доллара (_ и $). Имя может начинаться с любого допустимого символа, включая цифру. Имя не может включать символы прямого слэша (/), обратного слэша (\) и точки (.), не может состоять из одних цифр и не должно совпадать в написании с командами MySQL.

Тип_данных_столбца определяет тип значений, которые может принимать столбец. Спецификация типа также может определять максимальную длину значений, хранимых в столбце. Для некоторых типов длина определяется явным образом с указанием числа. В других случаях длина определяется именем типа. В качестве типов данных столбца можно выбрать любой тип из описанных в таблицах пунта 2.2.2.

После типа данных столбца определяются параметры (как необязательные параметры, зависящие от типа столбца, так и более общие параметры). Параметры действуют как модификаторы типов и вызывают изменение работы со столбцом.

Параметры столбцов

UNSIGNED (беззнаковый). Запрещает ввод отрицательных значений.

SIGNED (знаковый). Значения столбца могут принимать отрицательные значения. Параметр допустим только для целого типа данных. *По умолчанию* установлено значение SIGNED.

BINARY. Поле, чувствительное к регистру букв (для типов CHAR и VARCHAR).

NOT NULL. Определяет, что поле обязательно должно иметь значение при вставке новой записи в таблицу (если не задано значение по умолчанию). Если параметр NOT NULL не задан, то это означает, что поле может содержать пустые значения. *По умолчанию* установлено значение NULL.

DEFAULT значение_по_умолчанию. Задает для поля значение по умолчанию, которое будет использовано, если при вставке записи для этого поля не было явно указано значение. DEFAULT нельзя использовать для типа BLOB или TEXT.

Примечание – Если значение по умолчанию для столбца не задано (параметр DEFAULT отсутствует), но столбец объявлен как NOT NULL, то значение по умолчанию присваивается автоматически следующим образом:

- для числовых столбцов, кроме столбцов AUTO_INCREMENT, значение по умолчанию устанавливается равным 0. В столбце AUTO_INCREMENT значением по умолчанию является следующее значение в последовательности столбца;
- для типов даты и времени, кроме TIMESTAMP, по умолчанию устанавливается значение «0000-00-00»;
- для строковых типов, кроме типа ENUM, значением по умолчанию является пустая строка. Для типа ENUM таким значением является первый элемент перечня.

ZEROFILL (только для цифровых типов данных). Он указывает на то, что значения столбца при их отображении дополняются ведущими нулями до максимальной длины поля. Значения, длина которых превышает длину отображения, будут отображаться полностью без усечения.

Пример 1. Создадим в таблице mytbl столбец с именем nomer целого типа размерностью 5 знаков с параметром ZEROFILL:

```
CREATE TABLE mytbl (nomer INT(5) ZEROFILL);
```

Затем введем в него значения 1, 100, 1 000, 1 000 000 следующей командой:

```
INSERT INTO mytbl VALUES (1),(100),(1000),(1000000);
```

Тогда получим следующее содержимое столбца nomer: 00001, 00100, 01000, 1000000.

AUTO_INCREMENT. Этот параметр указывает, что поле является счетчиком. Обычно начальное значение равно 1 с автоматическим приращением 1.

Если при вставке новой записи указать NULL, то автоматически будет сгенерировано значение, на единицу большее максимального значения, уже существующего в поле.

Параметр AUTO_INCREMENT применяется только к столбцам, являющимся первичными ключами таблиц. Число столбцов AUTO_INCREMENT в таблице не может быть больше одного. Кроме того, так как порядковые номера всегда целые (INT) и положительные (UNSIGNED), их можно объявлять одновременно в описании столбца.

PRIMARY KEY. Указывает, что данный столбец является первичным ключом, содержащим только уникальные значения. PRIMARY KEY должен быть NOT NULL, т. е. не содержать пустых значений. Каждая таблица может иметь только один PRIMARY KEY.

UNIQUE. Определяет столбец как индекс UNIQUE, содержащий только уникальные значения. В отличие от PRIMARY KEY индекс UNIQUE может содержать несколько пустых значений. Индексов UNIQUE может быть несколько в одной таблице.

INDEX и KEY являются синонимами и определяют индексы, которые могут содержать повторяющиеся значения (используются для задания внешних ключей).

FULLTEXT. Индекс, который используется при осуществлении полнотекстового поиска или так называемого поиска по контексту. Индекс этого типа поддерживается только для таблиц типа MyISAM.

COMMENT 'строка комментария'. Определяет описательный комментарий, связанный со столбцом.

Дополнительные опции таблицы

AUTO_INCREMENT=число. Задает начальное значение для автоматической генерации значений столбца, заданного с типом AUTO_INCREMENT.

TYPE=тип_таблицы. Определяет структурный тип создаваемой таблицы. На практике обычно используются два типа таблиц – MyISAM и InnoDB.

DEFAULT CHARSET=кодировка. Определяет тип кодировки для создаваемой таблицы.

Рассмотрим примеры команд создания таблиц.

Пример 2. Напишем команду создания таблицы с именем primer. В состав таблицы входят: столбец kod, значения которого формируются автоматически с шагом 1, начиная с начального значения 10; а также два столбца с именами i1, i2 целого типа (типа INT) со значениями по умолчанию –1, 1:

```
CREATE TABLE primer
(kod INT UNSIGNED NOT NULL AUTO_INCREMENT,
i1 INT DEFAULT -1,
i2 INT DEFAULT 1
PRIMARY KEY (kod))
AUTO_INCREMENT=10
TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

Пример 3. Напишем команду создания таблицы mytbl, имеющей в своем составе столбцы kod, one, two и three.

Столбец kod может принимать только целые значения (INT), является первичным ключом, значения которого формируются автоматически с шагом 1, начиная с начального значения, равного 1.

Столбец one содержит значения вещественного типа FLOAT(5,2), цифра 5 указывает общее количество цифр в числе, а цифра 2 пока-

зывает количество цифр в дробной части. Это позволяет задавать значения столбца с точностью до сотых.

Значения столбца `two` являются строками символов длиной 15 `CHAR(15)`, не должны содержать пустые значения `NOT NULL`; по умолчанию в столбец записываются символы «пусто» (`DEFAULT "пусто"`).

Столбец `three` может принимать только целые значения `INT`, причем только положительные `UNSIGNED`, а также может содержать пустые значения `NULL`.

Команда `CREATE TABLE` для создания указанной выше таблицы будет иметь следующий вид:

```
CREATE TABLE mytbl
(kod INT UNSIGNED NOT NULL AUTO_INCREMENT,
one FLOAT(5,2),
two CHAR(15) NOT NULL DEFAULT "пусто",
three INT UNSIGNED
PRIMARY KEY (kod))
TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

2.2.2. Типы данных в столбцах таблицы

Типы данных, поддерживаемые СУБД MySQL, можно разделить на четыре группы: числа (Numbers), текст (Text), дата и время (Date and Time), списки (Defined Group).

Целые и вещественные типы данных

Система управления базами данных различает целые числа (например, 5 и 345) и вещественные числа (с дробной частью, например 123.5). Целые числа можно представить в двоичном или шестнадцатеричном формате. Также СУБД может работать с числами в экспоненциальной форме (например, 1.34E+14 или 2.53E–3). Целые и вещественные типы перечислены в таблице 1. Целые значения могут содержать положительные или отрицательные числа, а также 0. Таким образом, любой из этих типов может быть знаковым (`SIGNED`) или беззнаковым (`UNSIGNED`).

Таблица 1 – **Числовые типы столбцов**

Тип	Значение	Требуемая память
TINYINT[(m)]	Целое число от 0 до 255 для беззнаковых и от –128 до 127 для знаковых. Например: TINYINT(2) отображает значения от –9 до 99 для беззнаковых и от 0 до 99 для знаковых	1 байт
SMALLINT[(m)]	Целое число от 0 до 65 535 для беззнаковых и от –32 768 до 32 767 для знаковых	2 байта
INT[(m)]	Целое число от 0 до 4 294 967 295 для беззнаковых и от –2 147 683 648 до 2 147 483 647 для знаковых	4 байта
FLOAT[(m,d)]	Число с плавающей точкой обычной (одинарной) точности. Значения допустимы в пределах от –3.402 823 466 E+38 до –1.175 494 351 E–38 и от 1.175 4943 51 E–38 до 3.402 823 466 E+38. Например: поле типа FLOAT(7,2) будет отображать значения как 9 999.99	4 байта
Примечание – Параметр m обозначает общее количество знаков в числе, а d – количество знаков после запятой.		

Строковые типы данных

Строковые типы СУБД MySQL приведены в таблице 2. Текстовое поле может хранить любые символы, а также произвольные двоичные данные, такие как изображения и звуки.

Таблица 2 – **Строковые типы столбцов**

Тип	Значение	Требуемая память для хранения значения
CHAR(m)	Строка фиксированной длины (m принимает значения от 1 до 255 символов). Любой введенный текст меньшей длины будет дополнен пробелами в конце строки. Любой текст большей длины будет обрезан до заданной	m байт
VARCHAR(m)	Строка переменной длины (m принимает значения от 1 до 65 535 символов)	m+1 байт для записи длины строки
Примечание – Единственное различие между CHAR и VARCHAR заключается в том, что первый является типом с фиксированной длиной, а второй – с переменной. Все значения типа CHAR(m) занимают по m байт каждое, более короткие значения дополняются пробелами справа. Значения типа VARCHAR(m) занимают столько байт, сколько им необходимо, плюс 1 байт для хранения длины строки. При сохранении значений типа VARCHAR хвостовые пробелы отсекаются. В одной таблице нельзя смешивать столбцы типа CHAR и VARCHAR.		

Списки

В MySQL имеются два типа списков: ENUM (перечисление), SET (множество). Описание указанных типов приведено в таблице 3.

Таблица 3 – **Типы списков**

Тип	Значение	Требуемая память для хранения значения
Перечисление `ENUM ('значение1', 'значение2',...)`	Значения в столбце могут принимать только одно из текстовых значений набора. Набор может содержать до 65 535 различных элементов. Например: столбец, определенный как `ENUM ('один','два')` может принимать значения `'один'`, `'два'`	1 байт, если в наборе меньше 256 элементов, иначе – 2 байта
Множество `SET('значение1', 'значение2',...)`	Значения в столбце могут принимать несколько значений из набора. Набор может содержать до 64 различных элементов. Например: столбец, определенный как `SET('один','два')` может принимать значения `'один'`, `'два'`, `'один два'`	От 1 до 8 байт в зависимости от количества перечисленных элементов множества
Примечание – Тип `ENUM` желательно задавать вместе с параметром `NOT NULL`. Например, объявить столбец `pol` типом «перечень», принимающим значения «Муж» и «Жен», можно следующим образом: `pol ENUM ('МУЖ','ЖЕН') NOT NULL`.		

Типы данных для хранения даты и времени

Типы календарных данных, имеющиеся в СУБД MySQL, показаны в таблице 4. Здесь YY, MM, DD, hh, mm и ss соответствуют годам, месяцам, дням, часам, минутам и секундам.

Таблица 4 – **Столбцы календарного типа**

Тип	Значение
`DATE`	Дата в формате `"YYYY-MM-DD"`
`TIME`	Время в формате `"hh:mm:ss"`
`DATETIME`	Дата и время в формате `"YYYY-MM-DD hh:mm:ss"`
`YEAR [(2|4)]`	Значение года в формате `"YY"` или `"YYYY"`
Примечание – Даты должны задаваться в порядке год-месяц-день.	

2.2.3. Ключи и индексы

MySQL-таблица может иметь до 32 ключей и индексов, каждый из которых может иметь до 15 полей. Максимальная поддерживаемая длина ключа – 120 байт. Обратите внимание, что длинные ключи могут привести к низкой эффективности.

Ключи могут иметь имена. В случае первичного ключа имя будет всегда `PRIMARY KEY`. Если имя ключа не задано в процессе создания таблицы, то заданное по умолчанию имя ключа – первое имя столбца с факультативным суффиксом (_2, _3 и т. д.), чтобы сделать это имя уникальным.

2.2.4. Примеры создания таблиц

Пример 4. Создадим в базе данных `postavka` таблицу `product` с четырьмя столбцами: `kod` (код продукта), `name` (наименование продукта), `price` (цена продукта), `postav` (наименование поставщика). Столбец `kod` должен быть объявлен как первичный ключ `PRIMARY KEY`, а столбцы `name` и `postav` должны иметь индексы.

Можно воспользоваться командой `CREATE TABLE`, представленной в листинге 4.

Листинг 4. Команда создания таблицы `product`

```
USE postavka;
CREATE TABLE product
(kod INT NOT NULL AUTO_INCREMENT,
name CHAR(20) NOT NULL,
price FLOAT(5,1) NOT NULL,
postav CHAR(25) NOT NULL,
PRIMARY KEY (kod),
INDEX (name),
INDEX (postav))
TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

Примечание – Таблица `product` создается в базе данных `postavka`. Если база с таким именем еще не создана, то перед командой `USE` придется написать команду создания базы данных `CREATE DATABASE postavka;`

При создании ключа или индекса можно факультативно опреде-

лить, что только первые N символов поля будут использоваться под ключ или для создания индекса.

Пример 5. Для таблицы из предыдущего примера создадим индекс для поля name, в котором только первые пять символов из двадцати являются уникальными.

Напишем команду CREATE TABLE:

```
CREATE TABLE product
(kod INT NOT NULL AUTO_INCREMENT,
name CHAR(20),
price FLOAT(5,1),
postav CHAR(25),
PRIMARY KEY(kod),
INDEX name_idx(name(5)))
TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

Пример 6. Создадим таблицу studyFAM (где FAM – фамилия студента, набранная по-английски, например studyIvanov) в базе данных studentzaot.

Таблица должна содержать следующие данные о студентах:

- код студента kod (целое число, беззнаковое, пустые значения недопустимы, значения вводятся автоматически, является первичным ключом);
- фамилию студента fam (строка символов постоянной длины в 15 символов, пустые значения недопустимы, по полю создается индекс);
- имя студента im (строка символов постоянной длины в 15 символов, пустые значения недопустимы);
- пол pol (принимает одно из двух значений списка: «Ж» – женский, «М» – мужской);
- дату рождения data (тип «дата», пустые значения недопустимы);
- средний балл аттестата srbal (вещественное, с одной цифрой до запятой и одной цифрой после запятой).

Напишем команду CREATE TABLE:

```
USE studentzaot;
CREATE TABLE studyFAM
(kod INT UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
fam CHAR(15),
im CHAR(15),
pol ENUM('М','Ж')NOT NULL,
data DATE NOT NULL,
srbal FLOAT(3,1),
PRIMARY KEY(kod),
INDEX(fam))
TYPE=MyISAM;
```

2.2.5. Команды удаления таблиц и баз данных

Команда `DROP TABLE` предназначена для удаления одной или нескольких таблиц:

```
DROP TABLE имя_таблицы [,имя_таблицы, ...];
```

К примеру, для удаления таблицы `product` нужно выполнить SQL-запрос из листинга 5.

Листинг 5. Удаление таблицы

```
DROP TABLE product;
```

Команда `DROP DATABASE` удаляет базу данных со всеми таблицами, входящими в ее состав:

```
DROP DATABASE имя_базы_данных;
```

Так, удалить базу данных `postavka` можно SQL-запросом из листинга 6.

Листинг 6. Удаление базы данных

```
DROP DATABASE postavka;
```

Примечание – Прежде чем выполнять команды по удалению таблиц и баз данных, следует хорошо подумать, действительно ли необходимо это сделать.

Лабораторная работа 1
СОЗДАНИЕ ТАБЛИЦ В БАЗЕ ДАННЫХ

Порядок выполнения работы

1. Изучите команды создания базы данных, просмотра списка созданных баз данных и активизации (открытия) базы данных (подраздел 2.1).

2. Изучите синтаксис команды создания таблицы (подраздел 2.2).

3. Разберите примеры создания таблиц (примеры 1–6).

4. Самостоятельно напишите команду создания таблицы `productFAM` (где `FAM` – фамилия студента, набранная по-английски, например `productIvanov`) в базе данных `studentzaot`.

Таблица `productFAM` должна содержать пять столбцов (таблица 5):

- `kod` (код продукта) – целое, пустые значения недопустимы, значения вводятся автоматически, является первичным ключом;
- `name` (наименование продукта) – тип строковый, постоянной длины 25 символов, пустые значения недопустимы;
- `postav` (поставщик) – тип строковый, постоянной длины 30 символов, пустые значения недопустимы;
- `kol` (количество продукта) – целое, длина 7 знаков, пустые значения недопустимы;
- `price` (цена продукта) – целое, длина 10 знаков, пустые значения недопустимы.

Таблица 5 – **Записи таблицы `productFAM`**

Код	Наименование продукта	Поставщик	Количество	Цена

Примечание – Имена столбцов таблицы задавайте по-английски.

5. Сохраните описание таблицы `productFAM` в текстовый файл с именем *создание таблиц в mysql* (папка *STUD*). В него можете поместить синтаксис команд из заданий 1–2.

6. Самостоятельно напишите команду создания таблицы `klientFAM` (где `FAM` – фамилия студента, набранная по-английски, например `klientIvanov`) в базе данных `studentzaot`.

Таблица должна иметь структуру, представленную в таблице 6.

Таблица 6 – **Записи таблицы klientFAM**

Клиент	Статус	Email	Телефон	Город	Код страны	Страна

Требования к столбцам таблицы klientFAM:

- код клиента – целое число, беззнаковое, пустые значения недопустимы, значения вводятся автоматически;
- клиент – символьное, постоянной длины в 20 символов, пустые значения недопустимы;
- статус – принимает одно из двух значений списка: «Г» – государственная организация, «К» – коммерческая структура;
- email – символьное, постоянной длины в 10 символов;
- телефон – символьное, постоянной длины в 10 символов, пустые значения недопустимы;
- город – символьное, постоянной длины в 10 символов, пустые значения недопустимы;
- код страны – целое, 5 знаков;
- страна – символьное, постоянной длины в 20 символов, пустые значения недопустимы.

Поле код клиента является первичным ключом, по полю клиент создается индекс.

В отчет по лабораторной работе необходимо представить текстовый файл с именем *создание таблиц в mysql* в папке *STUD.*

2.3. Создание структуры таблицы с помощью программы mysqladmin

Программа mysqladmin кроме выполнения административных функций значительно упрощает процесс создания баз данных и таблиц в базе данных.

Однако прежде чем приступить к созданию базы данных или созданию таблицы в уже существующей базе данных, пользователю необходимо осуществить подключение к серверу баз данных MySQL.

Рассмотрим более подробно использование программы mysqladmin для установки соединения с сервером и создания таблиц.

2.3.1. Порядок действий по подключению к серверу баз данных MySQL

Обратимся к теории. Для подключения к серверу баз данных MySQL используется команда

```
mysql -h host_name -u user_name -p password;
```

где – h *host_name* указывает имя серверного узла (хоста), к которому следует подключиться и на котором находится сервер MySQL; если сервер баз данных работает на том же компьютере, что и программа-клиент, этот параметр всегда имеет имя `localhost` (и его можно не писать вообще);
`-u` *user_name* указывает имя пользователя (учетную запись пользователя баз данных), зарегистрированное в СУБД MySQL; по умолчанию (при первоначальной установке сервера MySQL) в базе данных имеется единственный пользователь с именем `root`;
`-p` *password* указывает пароль доступа к базе данных.

Итак, прежде чем стать пользователем баз данных MySQL, необходимо с помощью администратора баз данных зарегистрироваться на сервере MySQL и получить учетную запись с паролем доступа к серверу.

Допустим, студент получил от администратора учетную запись с именем `studentzaot` без пароля на учебном сервере университета, имеющем имя `uchserv`. Команда подключения к серверу будет иметь следующий вид:

```
mysql -h uchserv -u studentzaot -p;
```

А теперь воспользуемся программой mysqladmin для выполнения подключения к серверу баз данных:

1. Запустите программу mysqladmin, выполнив следующие команды:

Пуск → *Программы* → *MySQL* → *MySQL Administrator*.

В результате откроется окно задания параметров подключения к серверу MySQL (рисунок 2).

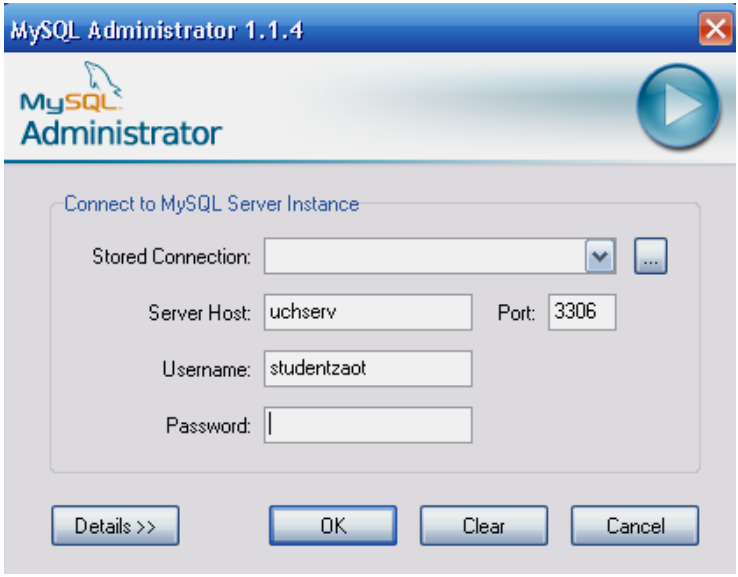

Рисунок 2 – **Стартовое окно программы mysqladmin**

2. Как видно из рисунка 2, пользователю необходимо указать:

• *имя серверного узла (Server Host)*, к которому необходимо подключиться;

• *имя пользователя (Username)*, зарегистрированное в СУБД MySQL;

• *пароль (Password)* доступа к базе данных.

Заполните указанные поля:

• в поле *Server Host* введите `uchserv`;

• в поле *Username* введите `studentzaot`.

Поле `Password` оставьте пустым.

Нажав кнопку *OK*, перейдем к основному окну программы с перечнем режимов работы (рисунок 3).

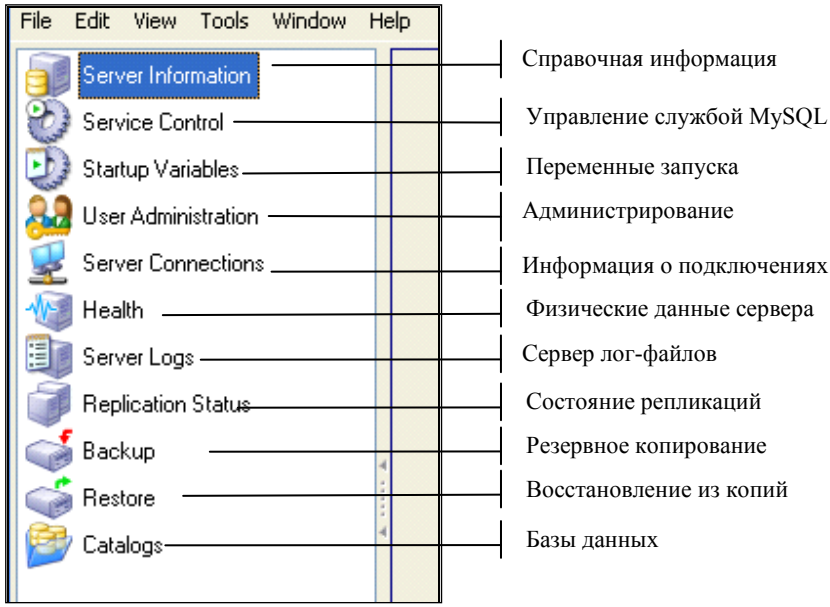

Рисунок 3 – **Режимы работы программы mysqladmin**

2.3.2. Создание таблицы в базе данных MySQL

Так как основное назначение изучаемой программы заключается в выполнении функций администрирования сервера, следовательно и большинство режимов предназначено для этих же целей (и доступно только пользователю, обладающему правами администратора сервера баз данных). Мы же для реализации своей задачи – создания таблицы в базе данных воспользуемся режимом *Catalogs*:

1. Щелкните кнопкой мыши по режиму *Catalogs*.

2. Из предложенного пользователю списка *имен баз данных*, с которыми ему разрешено работать (в левом нижнем углу), выберите базу данных `studentzaot` и выполните на ней щелчок мышью.

В результате будет получен доступ к *четырем вкладкам*, определяющим направления работы с выбранной базой данных:

- *Schema Tables* – просмотр списка таблиц, имеющихся в базе;
- *Schema Indices* – просмотр информации о ключах и индексах таблиц в выбранной базе данных;
- *Views* – просмотр;

• *Stored procedures* – создание сохраняемых процедур.

3. Сформулируем задачу: создать таблицу с именем `events№ауд №ПК` в базе данных `studentzaot` (например, `events2321`).

Требования к столбцам таблицы `events№ауд№ПК`:

• поле код события `id_event` – целое число, пустые значения недопустимы, значения вводятся автоматически, является первичным ключом таблицы;

• наименование события `name` – символьное, постоянной длины в 30 символов, пустые значения недопустимы;

• дата события `data` – тип «дата», пустые значения недопустимы.

Таким образом, столбцы таблицы будут описываться в следующем виде:

```
id_event INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(30) NOT NULL,
data DATE NOT NULL
```

4. Для создания таблицы откройте вкладку *Schema Tables* и нажмите кнопку *Create Table*. Откроется окно, представленное на рисунке 4.

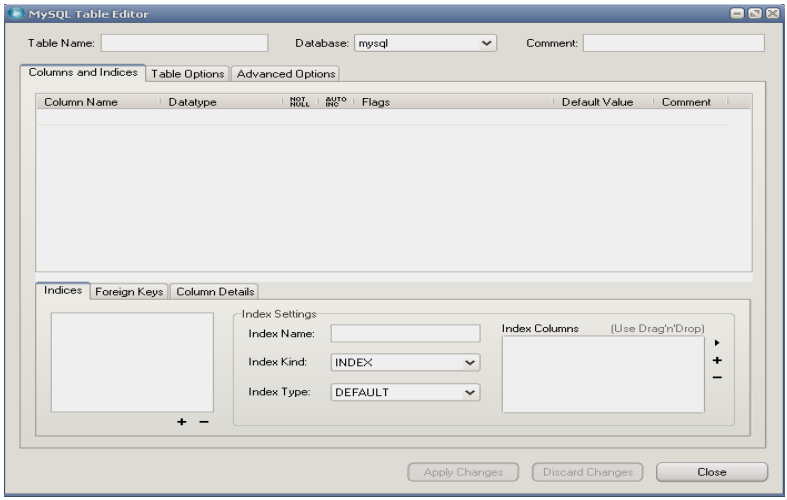

Рисунок 4 – **Стартовое окно создания таблицы в программе mysqladmin**

5. В поле *Table Name* введите имя создаваемой таблицы – `events№ауд№ПК`.

6. Далее щелкните на вкладке *Columns and Indices* и в поле *Column Name* после двойного щелчка мышью (или нажатия клавиши *Enter*) введите имя первого столбца таблицы – `id_event`.

После ввода имени столбца все его свойства заполняются программой автоматически. Пользователю необходимо только откорректировать выведенные значения свойств столбца.

Свойства столбца `id_event` должны быть заполнены в соответствии с первой строкой рисунка 5.

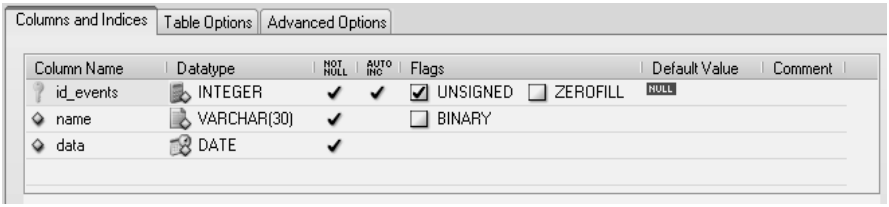

Рисунок 5 – **Полная информация по столбцу `id_event`**

Примечание – Для поля, являющегося первичным ключом таблицы (`id_event` в нашей таблице), нужно обязательно заполнить свойство поля *Default Value* значением `NULL`. Для этого щелкните кнопкой мыши на вкладке *Column Details*, а затем – по кнопке 

, расположенной правее поля *Default Value* (рисунок 6).

Рисунок 6 – **Свойства полей таблицы `events`**

7. Создайте еще два столбца таблицы `events№ауд№ПК` – `name` и `data`. Окончательный вариант перечня столбцов таблицы и описания их свойств должен иметь вид, соответствующий рисунку 5.

8. Просмотрите полную информацию о каждом созданном столбце, используя вкладку *Column Details* (см. рисунок 6).

9. Далее перейдите на вкладку *Table Options* и укажите:

• тип создаваемой таблицы `MyISAM`, поставив переключатель в соответствующее значение;

• кодировку cp1251 в поле *Charset*.

10. Сохраните структуру созданной таблицы, выполнив щелчок по кнопке *Apply Changes*.

В результате программой mysqladmin будет выдано окно, которое представляет команду создания таблицы `eventsNºаудNºПК` (рисунок 7). Изучите предлагаемую команду и определите, все ли параметры понятны.

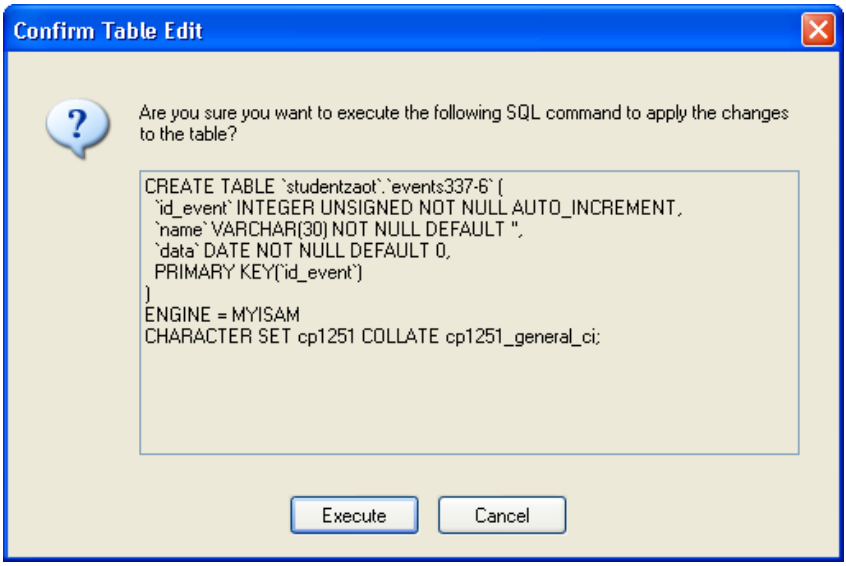

Рисунок 7 – **Полная информация по команде создания таблицы**

11. Подтвердите сохранение таблицы, нажав кнопку *Execute*.

12. Закройте окно создания таблицы кнопкой *Close*.

13. Удалите созданную таблицу, выделив ее в списке таблиц на вкладке *Schema Tables* и удалив командой *Drop* из контекстно-зависимого меню (рисунок 8).

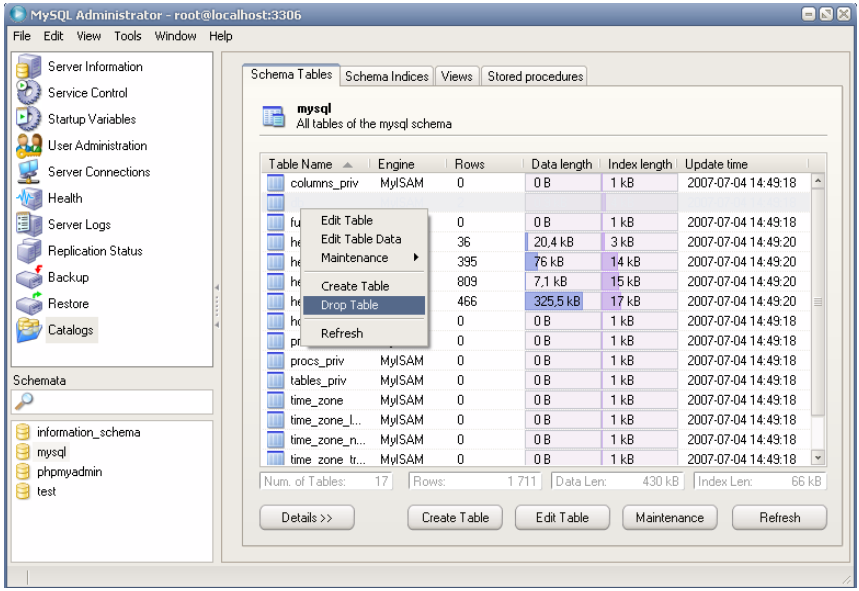

Рисунок 8 – **Удаление таблицы**

Лабораторная работа 2
СОЗДАНИЕ ТАБЛИЦ С ПОМОЩЬЮ ПРОГРАММЫ MYSQLADMIN

Порядок выполнения работы

1. Создайте таблицу с именем `eventsNaydNPK` в базе данных `studentzaot` (полное описание порядка ее создания содержится в пунктах 2.3.1–2.3.2). Параметры подключения к серверу баз данных уточните у преподавателя.
2. Удалите созданную вами таблицу.
3. С помощью программы mysqladmin в базе данных `studentzaot` (имя базы данных уточняйте у преподавателя) создайте таблицу `productFAM` (где `FAM` – ваша фамилия, набранная по-английски), описание которой приведено в лабораторной работе 1.
4. Результаты работы продемонстрируйте преподавателю.
5. С помощью программы mysqladmin в базе данных (имя базы данных уточните у преподавателя) создайте таблицу `klientFAM`.

Описание полей таблицы находится в пункте 6 лабораторной работы 1.

2.4. Команды работы с таблицами

Прежде чем рассмотреть примеры изменения структуры таблицы, создадим таблицу, с которой будем работать.

Пример 7. Создадим таблицу `study` в базе данных `studentzaot`.

Таблица должна содержать следующие данные о студентах:

- код студента `kod` (целое число, беззнаковое, пустые значения недопустимы, значения вводятся автоматически, является первичным ключом);
- фамилия студента `fam` (строка символов постоянной длины в 15 символов, по полю создается индекс);
- имя студента `im` (строка символов постоянной длины в 15 символов);
- пол `pol` (принимает одно из двух значений списка: «Ж» – женский, «М» – мужской, пустые значения недопустимы);
- дата рождения `data` (тип «дата», пустые значения недопустимы);
- средний балл аттестата `srbal` (вещественное, с одной цифрой до запятой и одной цифрой после запятой).

SQL-запрос для создания таблицы приведен в листинге 7.

Листинг 7. Создание таблицы `study` в базе данных

```
USE studentzaot;
CREATE TABLE study
(kod INT UNSIGNED NOT NULL AUTO_INCREMENT,
fam CHAR(15),
im CHAR(15),
pol ENUM('M','Ж')NOT NULL,
data DATE NOT NULL,
srbal FLOAT(3,1),
PRIMARY KEY (kod),
INDEX (fam))
TYPE=MyISAM;
```

2.4.1. Команда добавления записей в таблицу

Существует несколько методов добавления информации в базы данных: можно добавить записи в таблицы вручную с помощью оператора INSERT, можно добавлять записи прямо из текстового файла путем копирования в буфер, а можно готовый файл с исходными данными загрузить в таблицу с помощью оператора LOAD DATA. В этом разделе демонстрируется метод с использованием команды INSERT.

Команда INSERT вводит значения в записи таблицы:

```
INSERT INTO имя_таблицы VALUES (значения полей
1строки), (значения полей 2строки), (...),
(значения полей последней строки);
```

Примечание – Список VALUES должен содержать значения всех полей одной записи таблицы (обычно это порядок следования названий столбцов, заданный в операторе CREATE TABLE). Если необходимо узнать порядок следования столбцов, воспользуйтесь программой mysqladmin для просмотра перечня полей таблицы.

Вводить строковые значения и значения типа «дата» можно как в одинарных, так и двойных кавычках. Числовые поля задаются без использования кавычек.

Для того чтобы вставить в таблицу study из листинга 7 несколько записей с информацией о студентах, можно воспользоваться командой INSERT (листинг 8).

Листинг 8. Оператор INSERT

```
INSERT INTO study VALUES
(0,'Иванов','Михаил','М','1990-01-05',7.8),
(0,'Горовой','Максим','М','1991-01-03',5.2),
(0,'Максимова','Екатерина','Ж','1990-01-
03',9.0);
```

Примечание – Хотя первичный ключ (kod) в таблице study объявлен как AUTO_INCREMENT (автоматически нумеруемый системой), при вводе записей командой INSERT значение первичного ключа следует явно задавать равным нулю.

Порядок добавления столбцов можно задавать самостоятельно, воспользовавшись следующей формой оператора INSERT:

```
INSERT INTO имя_таблицы (имя_столбцаi, имя_
столбцаj) VALUES (значение_столбцаi, значение
столбцаj);
```

Например, запись в таблицу study можно осуществить при помощи SQL-запроса, представленного в листинге 9. Столбцы, не перечисленные в списке столбцов, получают значение по умолчанию. Так, первичный ключ получает значение NULL, которое интерпретируется для полей с атрибутом AUTO_INCREMENT генерацией уникального числа.

Листинг 9. Определение порядка добавления столбцов

```
INSERT INTO study (fam, pol, srbal) VALUES
('Смирнова','Ж',8.5);
```

MySQL позволяет задавать значения полей в операторе INSERT в форме имя_столбца=значение:

```
INSERT INTO имя_таблицы SET имя_столбцаi=
значение, имя_столбцаj=значение;
```

Пример такого оператора приведен в листинге 10.

Листинг 10. Альтернативная форма оператора INSERT

```
INSERT INTO study SET fam='Смирнова', pol='Ж';
```

Для полей, не получивших значения, будут выставлены значения по умолчанию.

2.4.2. Команда удаления записей из таблицы

Команда DELETE удаляет из таблицы записи, удовлетворяющие заданным условиям, и возвращает число удаленных записей:

```
DELETE FROM имя_таблицы [WHERE условие];
```

Важной частью запросов DELETE, UPDATE и SELECT является оператор WHERE, который позволяет задавать условия для выбора записей, на которые будут действовать эти команды. Запрос из листинга 11 удаляет из таблицы запись, значение первичного ключа в которой равно 2.

Листинг 11. Удаление записей по указанному критерию

```
DELETE FROM study WHERE kod=2;
```

Условия отбора могут быть значительно сложнее. Так, в листинге 12 удаляются все записи, в которых значение поля «пол» равно «мужской» и значение первичного ключа в которых превышает значение 3.

Листинг 12. Удаление записей со сложным критерием

```
DELETE FROM study WHERE pol='M' AND kod>3;
```

Оператор `AND` реализует логическое `И`. В запросах можно также применять логическое `ИЛИ` – `OR`.

Примечание – Система управления базами данных MySQL в качестве альтернативы операторам `AND` и `OR` поддерживает синтаксис, принятый в Си-подобных языках программирования, т. е. вместо `AND` можно применять `&&`, а вместо `OR` – `|` (вертикальную черту).

Удаление всех записей из таблицы `study` представлено в листинге 13.

Листинг 13. Удаление всех записей

```
DELETE FROM study;
```

2.4.3. Команда выборки записей из таблицы

Команда `SELECT` предназначена для извлечения строк данных из одной или нескольких таблиц и имеет в общем случае следующий синтаксис:

```
SELECT имя_столбца, ...
FROM имя_таблицы WHERE условие
[ORDER BY имя_столбца [ASC|DESC], ...]
[LIMIT [№записи,]количество_записей]
[GROUP BY имя_столбца[,имя столбца]];
```

Здесь имя_столбца – имя выбираемого столбца. Можно указать несколько столбцов через запятую, а если необходимо выбрать все столбцы, можно просто ввести символ «звездочка» (*). После ключевого слова FROM указывается имя таблицы, из которой извлекаются записи. Ключевое слово WHERE определяет, так же как и в операторе DELETE, условия отбора строк. Ключевое слово ORDER BY сортирует строки результата по столбцу имя_столбца в прямом (ASC) или обратном порядке (DESC). Ключевое слово LIMIT сообщает MySQL о выводе такого количества записей, которое задано в параметре количество_записей после записи, номер которой задан в параметре №записи (нумерация записей начинается с нуля). Ключевое слово GROUP BY задает условие группировки записей.

Для применения команды выборки записей из базы данных при помощи оператора SELECT воспользуемся ранее созданной таблицей studyFAM в базе данных studentzaot (см. листинг 7) и добавим в нее несколько записей (листинг 14).

Листинг 14. Вставка пяти записей в таблицу study

```
INSERT INTO study VALUES
 (0,'Петров','Михаил','М','1990-15-05',6.8),
 (0,'Горовая','Марина','Ж','1991-04-02',8.2),
 (0,'Семенова','Екатерина','Ж','1991-01-
03',9.0),
 (0,'Карась','Николай','М','1990-01-05',7.8),
 (0,'Горкин','Максим','М','1991-01-03',5.2);
```

Для того чтобы посмотреть содержание всей таблицы, выполняется запрос из листинга 15. Значения всех столбцов возвращаются в том же порядке, в котором они хранятся в таблице. Этот порядок совпадает с порядком, в котором столбцы перечислены оператором DESCRIBE.

Листинг 15. Команда SELECT

```
SELECT * FROM study;
```

В данном запросе происходит выборка всех столбцов из таблицы study без ограничений. Результат запроса показан на рисунок 9.

```
mysql> select * from study;
+-----+----------+-----------+-----+------------+-------+
| kod | fam      | im        | pol | data       | srbal |
+-----+----------+-----------+-----+------------+-------+
|   1 | Петров   | Михаил    | М   | 1990-05-05 |   6.8 |
|   2 | Горовая  | Марина    | Ж   | 1991-04-02 |   8.2 |
|   3 | Семенова | Екатерина | Ж   | 1991-01-03 |   9.0 |
|   4 | Карась   | Николай   | М   | 1990-01-05 |   7.8 |
|   5 | Горкин   | Максим    | М   | 1991-01-03 |   5.2 |
+-----+----------+-----------+-----+------------+-------+
5 rows in set (0.00 sec)
```

Рисунок 9 – **Результат выполнения запроса из листинга 15**

Выборка из определенных полей

Можно выбрать не все столбцы таблицы, а лишь часть, для этого необходимо явно задать список выбираемых столбцов (листинг 16).

Листинг 16. Частичная выборка

```
SELECT kod, fam FROM study;
```

В этом случае MySQL выведет лишь два столбца с первичным ключом `kod` и фамилией студента `fam` (рисунок 10).

```
mysql> SELECT kod, fam FROM study;
+-----+----------+
| kod | fam      |
+-----+----------+
|   1 | Петров   |
|   2 | Горовая  |
|   3 | Семенова |
|   4 | Карась   |
|   5 | Горкин   |
+-----+----------+
5 rows in set (0.00 sec)
```

Рисунок 10 – **Результат выполнения запроса из листинга 16**

Ограничение количества строк результатов запроса

Ключевое слово `LIMIT` используется для ограничения количества строк, возвращаемых командой `SELECT` (листинг 17).

Листинг 17. Использование ключевого слова `LIMIT` в операторе `SELECT`

```
SELECT * FROM study LIMIT 3;
```

В результате запроса будут выведены только первые три записи из пяти.

`LIMIT` может также принимать два целочисленных аргумента. В этом случае первый аргумент сообщает MySQL, *с какой строки* производить отсчет, а второй аргумент задает максимальное *количество возвращаемых строк* (листинг 18).

Листинг 18. Альтернативная форма задания предложения `LIMIT`

```
SELECT * FROM study LIMIT 1,3;
```

В этом случае будут возвращены строки 2, 3 и 4 (рисунок 11).

```
mysql> SELECT * FROM study LIMIT 1,3;
+-----+----------+-----------+-----+------------+-------+
| kod | fam      | im        | pol | data       | srbal |
+-----+----------+-----------+-----+------------+-------+
|   2 | Горовая  | Марина    | Ж   | 1991-04-02 |   8.2 |
|   3 | Семенова | Екатерина | Ж   | 1991-01-03 |   9.0 |
|   4 | Карась   | Николай   | М   | 1990-01-05 |   7.8 |
+-----+----------+-----------+-----+------------+-------+
3 rows in set (0.00 sec)
```

Рисунок 11 – **Результат выполнения запроса из листинга 18**

Определение критериев выбора записей

Ограничение набора выбираемых командой `SELECT` записей производится с помощью предложения `WHERE`, которым определяется набор выбираемых строк.

В качестве критериев можно задавать цифровые значения, значения типа `DATA`, а также комбинации значений и даже арифметические операторы. Оператор `WHERE` применяется в команде `SELECT` точно так же, как и в команде `DELETE`. Выберем из таблицы `study` только те записи, у которых значение `kod` больше 2 и меньше 5 (листинг 19).

Листинг 19. Задание условий в команде SELECT

```
SELECT * FROM study WHERE kod>2 and kod<5;
```

Результат листинга показан на рисунке 12.

```
mysql> SELECT * FROM study WHERE kod>2 and kod<5;
+-----+----------+-----------+-----+------------+-------+
| kod | fam      | im        | pol | data       | srbal |
+-----+----------+-----------+-----+------------+-------+
|   3 | Семенова | Екатерина | Ж   | 1991-01-03 |   9.0 |
|   4 | Карась   | Николай   | М   | 1990-01-05 |   7.8 |
+-----+----------+-----------+-----+------------+-------+
2 rows in set (0.03 sec)
```

Рисунок 12 – **Результат выполнения запроса из листинга 19**

Сортировка результатов запроса

Порядок сортировки выводимых записей можно задавать при помощи оператора `ORDER BY` (листинг 20).

Листинг 20. Сортировка результатов запроса

```
SELECT * FROM study WHERE kod>2 ORDER BY data;
```

В этом запросе выводятся все записи со значением поля `kod` больше 2, которые при этом сортируются по значению поля `data` (рисунок 13).

```
mysql> SELECT * FROM study WHERE kod>2 ORDER BY data;
+-----+----------+-----------+-----+------------+-------+
| kod | fam      | im        | pol | data       | srbal |
+-----+----------+-----------+-----+------------+-------+
|   4 | Карась   | Николай   | М   | 1990-01-05 |   7.8 |
|   3 | Семенова | Екатерина | Ж   | 1991-01-03 |   9.0 |
|   5 | Горкин   | Максим    | М   | 1991-01-03 |   5.2 |
+-----+----------+-----------+-----+------------+-------+
3 rows in set (0.00 sec)
```

Рисунок 13 – **Результат выполнения запроса из листинга 20**

Ключевым словом `ASC` или `DESC` можно задать порядок сортировки столбца по возрастанию или убыванию. Например, для отображения списка студентов в порядке убывания их порядкового номера (поле `kod`) можно сделать запрос в соответствии с листингом 21, результат

выполнения которого можно посмотреть на рисунке 14.

Листинг 21. Сортировка результатов запроса по убыванию

```
SELECT * FROM study ORDER BY kod DESC;
```

```
mysql> SELECT * FROM study ORDER BY kod DESC;
+-----+----------+-----------+-----+------------+-------+
| kod | fam      | im        | pol | data       | srbal |
+-----+----------+-----------+-----+------------+-------+
|   5 | Горкин   | Максим    | М   | 1991-01-03 |   5.2 |
|   4 | Карась   | Николай   | М   | 1990-01-05 |   7.8 |
|   3 | Семенова | Екатерина | Ж   | 1991-01-03 |   9.0 |
|   2 | Горовая  | Марина    | Ж   | 1991-04-02 |   8.2 |
|   1 | Петров   | Михаил    | М   | 1990-05-05 |   6.8 |
+-----+----------+-----------+-----+------------+-------+
5 rows in set (0.00 sec)
```

Рисунок 14 – **Результат выполнения запроса из листинга 21**

Сравнение по шаблону

В СУБД MySQL реализовано сравнение по шаблонам:

`LIKE 'шаблон'` – значение поля соответствует шаблону;
`NOT LIKE 'шаблон'` – значение поля не соответствует шаблону.

При описании можно использовать символы шаблона:
`'_'` (нижнее подчеркивание) – соответствует любому одиночному символу;
`'%'` (процент) – соответствует любому количеству символов.

В MySQL шаблоны по умолчанию не чувствительны к регистру символов.

Так SQL-запрос к таблице `study`, сформированный в листинге 22, приводит к выводу списка фамилий студентов мужского пола, имя которых начинается на букву «М». Отображаемый список содержит только три столбца: фамилия, имя, пол (рисунок 15).

Листинг 22. Применение оператора `SELECT` с шаблоном

```
SELECT fam, im, pol FROM study WHERE im LIKE
'M%' and pol='M';
```

```
mysql> SELECT fam, im, pol FROM study WHERE im LIKE 'M%' and pol='M';
+--------+--------+-----+
| fam    | im     | pol |
+--------+--------+-----+
| Петров | Михаил | M   |
| Горкин | Марк   | M   |
+--------+--------+-----+
2 rows in set (0.00 sec)
```

Рисунок 15 – **Результат выполнения SQL-запроса из листинга 22**

Получение итоговых результатов

Функция `COUNT` позволяет посчитать количество строк, выбранных запросом. Например, SQL-запрос в листинге 23 позволяет выяснить общее количество студентов (рисунок 16).

Листинг 23. Общее количество строк в таблице `study`

```
SELECT COUNT(*) FROM study;
```

```
mysql> SELECT COUNT(*) FROM study;
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.02 sec)
```

Рисунок 16 – **Результат выполнения SQL-запроса из листинга 23**

В качестве аргумента функции может выступать имя столбца таблицы. Запрос, приведенный ниже, аналогичен по результату запросу из листинга 24.

Листинг 24. Альтернативный оператор `SELECT`

```
SELECT COUNT(fam) FROM study;
```

Функция `COUNT(*)` подсчитывает каждую выбранную строку. Функция `COUNT(fam)` подсчитывает строки, содержащие ненулевые значения только в столбце `fam`.

Можно подводить итоги по отдельным категориям. Например, общее число студентов мужского пола можно определить с помощью

запроса, представленного в листинге 25.

Листинг 25. Общее количество студентов мужского пола

```
SELECT COUNT(*) FROM study WHERE pol='M';
```

Кроме `COUNT()`, существуют и другие агрегирующие функции. Функции `MIN()`, `MAX()`, `SUM()` и `AVG()` предназначены для вычисления минимума, максимума, суммы и среднего значения столбца соответственно. Их можно использовать одновременно. Запрос, определяющий минимальное и максимальное значение среднего балла аттестата, представлен в листинге 26 (рисунок 17).

Листинг 26. Использование функций `MIN()`, `MAX()`

```
SELECT MIN(srbal) AS minimum, MAX(srbal) AS max-
imum FROM study;
```

```
mysql> SELECT MIN(srbal) AS minimum, MAX(srbal) AS maximum FROM study;
+---------+---------+
| minimum | maximum |
+---------+---------+
|     5.2 |     9.0 |
+---------+---------+
1 row in set (0.00 sec)
```

Рисунок 17 – **Результат выполнения SQL-запроса из листинга 26**

Группировка записей таблицы

Наиболее продуктивно функция `COUNT()` используется совместно с оператором группировки `GROUP BY`. Так, подсчет количества студентов отдельно мужского и женского пола можно осуществить одним SQL-запросом, приведенным в листинге 27.

Листинг 27. Совместное использование `COUNT()` и `GROUP BY`

```
SELECT pol, COUNT(*) AS vsego FROM study GROUP
BY pol;
```

Результат работы запроса представлен на рисунке 18, из которого видно, что в таблицу внесено три студента мужского пола и два – женского.

```
mysql> SELECT pol, COUNT(*) AS vsego FROM study GROUP BY pol;
+-----+-------+
| pol | vsego |
+-----+-------+
| М   |     3 |
| Ж   |     2 |
+-----+-------+
2 rows in set (0.00 sec)
```

Рисунок 18 – **Результат выполнения SQL-запроса из листинга 27**

Исключение дубликатов

Часто данные в таблицах дублируются. Получить выборку уникальных значений позволяет ключевое слово `DISTINCT`, которое помещается перед именем столбца. В листинге 28 приводится выборка неповторяющихся значений по столбцу `pol`.

Листинг 28. Использование ключевого слова `DISTINCT`

```
mysql> SELECT DISTINCT pol FROM study;
```

Результат листинга представлен на рисунке 19.

```
mysql> SELECT DISTINCT pol FROM study;
+-----+
| pol |
+-----+
| М   |
| Ж   |
+-----+
2 rows in set (0.00 sec)
```

Рисунок 19 – **Результат выполнения SQL-запроса из листинга 28**

Ключевое слово `DISTINCT` может использоваться совместно с агрегирующей функцией `COUNT`. Все студенты, занесенные в таблицу `study`, должны иметь уникальные порядковые номера (неповторяющиеся коды студента). Проверить выполнение этого требования можно при помощи следующего SQL-запроса:

```
SELECT COUNT(DISTINCT kod) FROM study;
```

Если полученный результат не совпадает с результатом работы листинга 23, то это значит, что предъявленное выше требование не

соблюдается, т. е. в таблице имеются студенты с одинаковым кодом.

2.4.4. Команда обновления значений столбцов

Команда `UPDATE` обновляет содержимое столбцов таблицы в соответствии с их новыми значениями:

```
UPDATE имя_таблицы
SET имя_столбца=новое_значение
[,имя_столбца=новое_значение  ...]
[WHERE условие]
[LIMIT количество_записей];
```

В выражении `SET` указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении `WHERE`, если оно присутствует, задается условие отбора строк для обновления. В остальных случаях обновляются все строки. Ключевое слово `LIMIT` позволяет ограничить число обновляемых строк.

В листинге 29 для студента с фамилией Горкин меняется имя с Максима на Марк. Результат выполнения запроса представлен на рисунке 20.

Листинг 29. Применение оператора `UPDATE`

```
UPDATE study SET im='Марк' WHERE fam='Горкин';
```

```
+-----+----------+-----------+-----+------------+-------+
| kod | fam      | im        | pol | data       | srbal |
+-----+----------+-----------+-----+------------+-------+
|   1 | Петров   | Михаил    | М   | 1990-05-05 |   6.8 |
|   2 | Горовая  | Марина    | Ж   | 1991-04-02 |   8.2 |
|   3 | Семенова | Екатерина | Ж   | 1991-01-03 |   9.0 |
|   4 | Карась   | Николай   | М   | 1990-01-05 |   7.8 |
|   5 | Горкин   | Марк      | М   | 1991-01-03 |   5.2 |
+-----+----------+-----------+-----+------------+-------+
5 rows in set (0.00 sec)
```

Рисунок 20 – **Результат выполнения SQL-запроса из листинга 28**

Лабораторная работа 3
РАБОТА С ТАБЛИЦАМИ БАЗЫ ДАННЫХ

Порядок выполнения работы

1. Напишите вручную SQL-команду добавления одной записи в таблицу `productFAM` в базе данных `studentzaot` в соответствии с данными таблицы 7.

Таблица 7 – **Таблица `productFAM` базы данных `studentzaot`**

Код	Наименование продукта	Поставщик (страна)	Количество	Цена
1	Яблоки	Молдавия	100	3 100

2. Напишите вручную SQL-команду удаления всех записей из таблицы `productFAM` в базе данных `studentzaot`.

3. Для выполнения операций добавления записей в таблицу, а затем – выборки записей желательно воспользоваться программным приложением heidisql.exe. Указанная программа предлагает удобный графический интерфейс работы с таблицами базы данных MySQL.

4. Запустите файл heidisql.exe. В стартовом окне заполните параметры подключения к серверу MySQL в соответствии с рисунком 21. Затем нажмите кнопку *Open*.

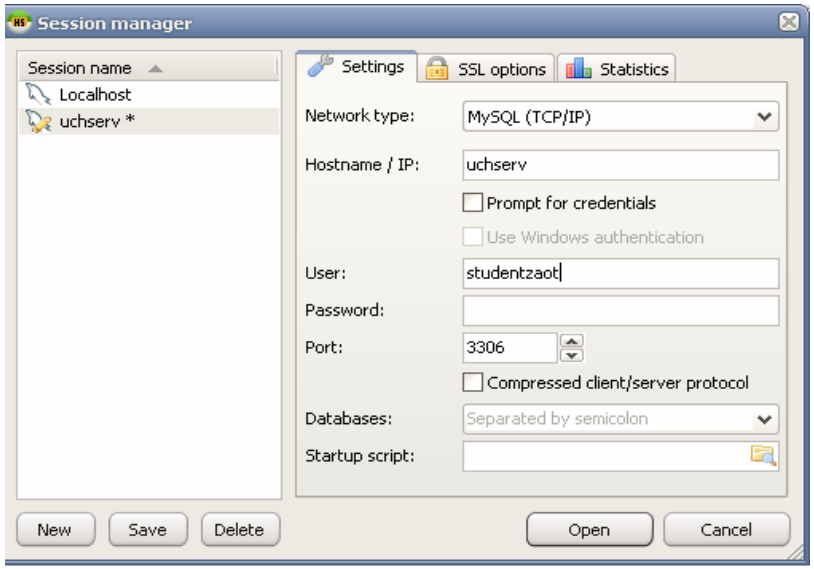

Рисунок 21 – **Стартовое окно программы heidisql.exe**

5. В списке баз данных, расположенном в левой части окна программы, найдите имя базы `studentzaot` и щелкните по значку «+». Раскроется список таблиц, содержащихся в выбранной базе данных.

6. Выделите свою таблицу, созданную в лабораторной работе 1 (`productFAM`) и нажмите кнопку *Data*, расположенную вверху правой части окна. Данный режим предназначен для ввода записей в таблицу.

7. Вызовите контекстное меню в свободном месте правой части окна и выберите в нем команду *Insert row* (*Ввод строки*). Введите значения первой записи.

8. Затем введите вторую запись, вновь вызвав контекстное меню и команду *Insert row*. Введите все семь записей в соответствии с таблицей 8.

Таблица 8 – **Записи таблицы `productFAM`**

Код	Наименование продукта	Поставщик (страна)	Количество	Цена
1	Яблоки	Молдавия	100	3 100
2	Груши	Италия	50	4 500
3	Персики	Испания	120	5 600
4	Яблоки	Молдавия	180	3 400
5	Яблоки	Испания	200	5 400
6	Яблоки	Италия	200	3 800
7	Мандарины	Испания	500	12 000

9. Теперь можно приступить к созданию запросов – отбору записей из таблицы по определенным условиям. Для создания запросов необходимо перейти в режим ввода запросов, нажав кнопку *Query*.

10. Приступите к набору SQL-запроса. Для облегчения ввода стандартных параметров запроса можно воспользоваться ключевыми словами, раскрыв список команд категории *SQL Keywords* (*ключевые слова SQL-запросов*) (рисунок 22).

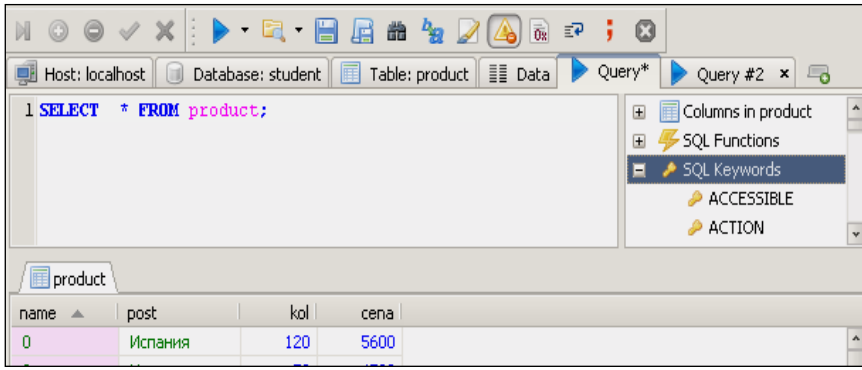

Рисунок 22 – **Окно запроса в программе heidisql.exe**

11. Текст запроса можно выполнить, нажав клавишу *F9* или выбрав команду выполнения запроса *Run*, раскрыв вложенный список кнопки ▶ (рисунок 23).

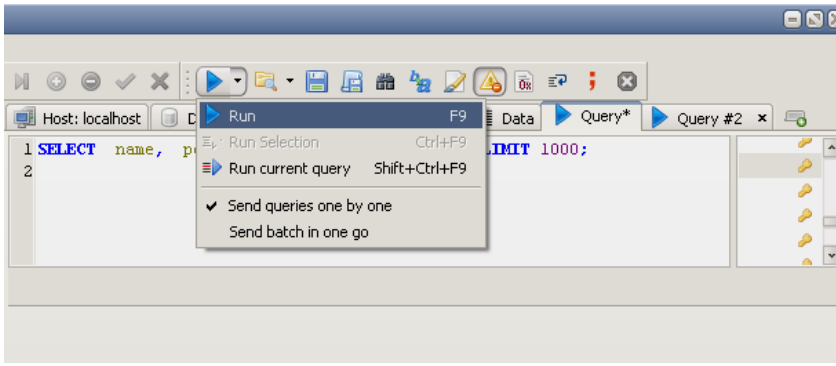

Рисунок 23 – **Запуск запроса на выполнение в окне программы heidisql.exe**

12. Посмотрите результат выполнения запроса, представленный в окне программы.

13. Задайте SQL-команды выборки записей из таблицы `productFAM` в базе данных `studentzaot` в соответствии с условиями, указанными ниже:

13.1. Выберите все записи, которые имеются в таблице.

13.2. Извлеките и отобразите содержимое только двух столбцов: «Наименование продукта» и «Цена».

13.3. Отобразите только две первые записи из таблицы.

13.4. Выберите все записи по продуктам, наименование которых начинается на букву «Я» или «Г».

13.5. Выберите все записи по странам, наименование которых начинается с буквы «И».

13.6. Выберите все записи, в которых «Цена» больше 3 000 и меньше 5 000, отсортировав их по убыванию цены. Отобразите результат только по столбцам «Наименование продукта» и «Цена».

13.7. Рассчитайте общее количество продуктов каждого наименования. В результате отобразите только два столбца: «Наименование продукта» и «Итог по количеству».

13.8. Рассчитайте общее число поставщиков каждого вида продуктов. В результате отобразите два столбца: «Наименование продукта» и «Итого».

13.9. Какая из стран является лидером по количеству поставленного товара? Создайте запрос, отвечающий на поставленный вопрос.

13.10. Рассчитайте общее число поставщиков. В результате должен находиться только один столбец с итоговой цифрой количества поставщиков.

Примечание – Для сохранения текстов запросов в файл воспользуйтесь командой *Save as*, вызвав ее из контекстного меню указанного запроса.

3. ПРИЕМЫ РАБОТЫ С SQL-СЕРВЕРОМ СРЕДСТВАМИ PHP

3.1. Принципы работы с базой данных через веб-интерфейс

Порядок работы с базой данных на MySQL-сервере через веб-интерфейс, организованный средствами PHP, выглядит примерно следующим образом (рисунок 24):

1. Пользователь в адресной строке веб-браузера задает имя открываемой HTML-страницы.

2. Веб-сервер выдает пользователю HTML-страницу с формой, в которой пользователь может сформулировать запрос к базе данных.

3. Веб-сервер принимает запрос, открывает соответствующий PHP-сценарий и передает его на обработку механизму PHP.

4. Модуль PHP приступает к разбору сценария. Сценарий содержит команду открытия соединения с базой данных и текст запроса.

5. Механизм PHP открывает соединение с MySQL-сервером и перенаправляет соответствующий запрос серверу MySQL.

6. Сервер MySQL принимает запрос к базе данных, обрабатывает его и отправляет результат (выборку из базы данных) обратно меха-

низму PHP.

7. Механизм PHP завершает выполнение сценария, что обычно включает в себя форматирование результатов запроса в HTML. После этого результат в формате HTML возвращается веб-серверу.

8. Веб-сервер пересылает HTML-документ в браузер, и пользователь имеет возможность просмотреть результаты своего запроса к базе данных.

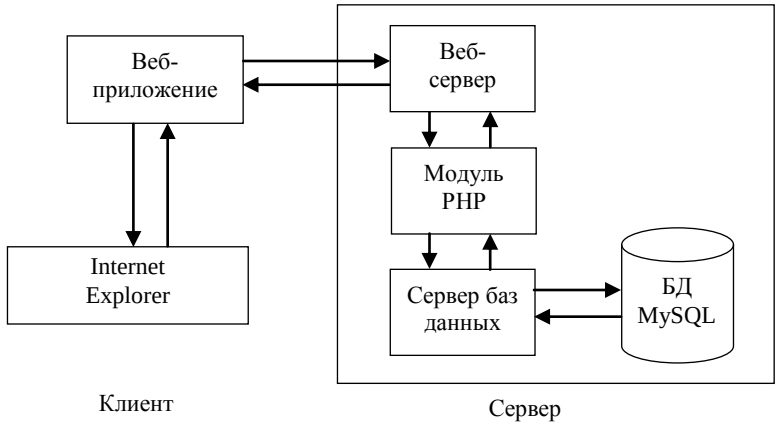

Рисунок 24 – **Реализация работы с базами данных через веб-интерфейс**

3.2. PHP-функции для работы с сервером баз данных

Установка соединения с сервером

Первым шагом при работе с SQL-сервером средствами PHP является установка соединения с ним. Это осуществляется при помощи функции `mysql_connect()`, которая имеет следующий синтаксис:

```
mysql_connect("server","username", "password")or
die ("Ошибка соединения с сервером");
```

Примечание – Все аргументы функции являются необязательными. В случае их отсутствия по умолчанию для этой функции устанавливаются следующие параметры: `server`='localhost:3306', `username` принимает значение владельца сервера, а `password` – пустая строка.

В случае успеха функция возвращает дескриптор соединения[1] с сервером, при неудаче – возвращает значение `false`.

Пример 8. Рассмотрим пример соединения с сервером MySQL, установленным на учебном сервере университета. Допустим, вы получили учетную запись с именем `studentzaot` без пароля на учебном сервере университета, имеющем имя `uchserv`. Тогда придется написать PHP-сценарий, представленный в листинге 30.

Листинг 30. Параметры подключения к серверу (файл config.php)

```
<?php
// выполняется подключение к серверу
$connect=mysql_connect("uchserv",
"studentzaot","");
// задается кодировка по умолчанию
mysql_query("set names cp1251");
?>
```

Примечание – Комментарии, которые в тексте листинга записывают после символов /* или //, можно не писать.

Рассмотренный выше сценарий из листинга 30 можно сохранить в файл с именем `config.php`. В дальнейшем, при написании новых скриптов на PHP, для подключения к серверу баз данных можно будет воспользоваться уже сохраненным файлом `config.php`. Включение существующего файла в новый скрипт выполняется с помощью команды `include` (листинг 31).

Листинг 31. Включение одного сценария в другой

```
<?php
include "config.php";
?>
```

[1] *Дескриптор соединения* представляет собой указатель на открытое соединение клиента с сервером, который используется для работы с несколькими подключениями.

Функция закрытия соединения с SQL-сервером

После завершения работы с базой данных необходимо разорвать соединение с SQL-сервером с помощью функции `mysql_close()`, имеющей следующий синтаксис:

```
mysql_close([дескриптор соединения]);
```

В качестве необязательного параметра функция принимает `дескриптор открытого соединения`. Если этот параметр не указан, закрывается последнее открытое соединение. Функция возвращает `true` в случае успеха и `false` – при возникновении ошибки.

В листинге 32 приведен пример использования функции.

Листинг 32. Функция `mysql_close()`

```
<?php
// устанавливаем соединение с сервером
$name_server="uchserv";
$user="studentzaot";
$pass="";
$connect=mysql_connect($name_server, $user, $pass);
// выбираем базу данных
$dbname="studentzaot";
mysql_select_db($dbname, $connect);
// закрываем соединение с сервером
mysql_close($connect);
?>
```

Функция указания имени открываемой базы данных

После того как соединение установлено, необходимо выбрать базу данных, с которой пользователь желает работать. Это осуществляется при помощи функции `mysql_select_db()`, которая имеет следующий синтаксис:

```
mysql_select_db(имя_базы_данных[,дескриптор
соединения]);
```

Использование этой функции эквивалентно вызову команды `USE` в SQL-запросе, т. е. функция `mysql_select_db()` выбирает базу

данных для дальнейшей работы, и все последующие SQL-запросы применяются к выбранной базе данных. Функция возвращает `true` при успешном выполнении операции и `false` в противном случае.

Имеет смысл помещать функции соединения с сервером и выбора базы данных в один и тот же файл (`config1.php`), где объявлены переменные с именами сервера, пользователя и паролем (листинг 33).

Листинг 33. Файл `config1.php`

```
<?php
// подключение к серверу
$connect=mysql_connect("uchserv", "studentzaot",
"");
// запрос на открытие базы studentzaot
mysql_select_db("studentzaot", $connect);
// задается кодировка по умолчанию
mysql_query("set names cp1251");
?>
```

Примечание – Комментарии, которые в тексте листинга записывают после символа //, можно не писать.

3.3. Основные PHP-функции для работы с таблицами базы данных

Функция создания SQL-запросов

После установки соединения с сервером баз данных и указания имени используемой базы выполняются SQL-запросы к базе данных из PHP-сценариев при помощи функции `mysql_query()`. Она имеет следующий синтаксис:

```
mysql_query(SQL-запрос[,дескриптор соединения])
or die("Ошибка: ".mysql_error());
```

Первый аргумент функции (`SQL-запрос`) представляет собой строку с SQL-запросом, второй (`дескриптор соединения`) представляет собой указатель соединения, возвращаемый функцией `mysql_connect()`.

Пользователь, работающий с сервером баз данных, расположенным на учебном сервере университета, не обладает правами на создание баз данных.

Для создания таблиц в уже имеющейся базе данных лучше воспользоваться специальной программой-администратором mysqladmin (подраздел 2.3).

С помощью указанной программы была создана таблица `productFAM` в базе данных `studentzaot` (лабораторная работа 2).

Рассмотрим более подробно написание сценария на PHP для добавления записей в таблицу.

Пример 9. Добавление данных. Созданная таблица `productFAM` пуста. Чтобы наполнить ее данными, можно воспользоваться запросом с SQL-командой `INSERT`. После установки соединения с SQL-сервером и выбора базы данных формируется запрос на добавление строки в таблицу, который исполняется при помощи уже знакомой функции `mysql_query()`.

В листинге 34 в таблицу `productFAM` добавляются три строки, содержащие сведения про яблоки, груши и персики.

Листинг 34. Добавление данных в таблицу (файл insert_table.php)

```
<HTML>
<BODY>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
// формирование SQL-запроса на добавление записей в таблицу
$query="INSERT INTO productFAM VALUES
(0,'яблоки','Молдавия',100,3100),(0,'груши',
'Италия',50,4500),(0,'персики','Испания',120,
5600)";
// в переменную result заносится результат запроса
$result=mysql_query($query)
or die ("Ошибка запроса: ".mysql_error());
echo "Данные успешно добавлены ";
// закрытие соединения
mysql_close($connect);
?>
```

```
</BODY>
</HTML>
```

Примечание – Добавление данных в таблицу удобнее выполнять из HTML-форм с передачей параметров методом POST (подраздел 3.4).

Функция для отображения результатов SQL-запросов

В предыдущем материале была создана таблица `productFAM`, содержащая три строки. Чтобы выполнить выборку и отображение всех записей из этой таблицы, выполним SQL-запрос `SELECT`. В результате выполнения этого запроса сформируется массив[1], содержащий все строки требуемой таблицы:

```
$query = "SELECT * FROM productFAM";
$result = mysql_query($query)
or die ("Ошибка при выполнении запроса:
".mysql_error());
```

Для организации последовательного отображения всех строк таблицы используется цикл `while` в сочетании с функцией `mysql_fetch_array()`. Эта функция возвращает одну очередную строку из результирующего набора данных в виде массива, проиндексированного именами полей таблицы. Синтаксис данной функции:

```
mysql_fetch_array(результат запроса)
```

В качестве аргумента `результат запроса` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

Возвращаемый массив содержит значения, идентифицируемые как по числовым индексам столбцов (численный массив), так и именами столбцов (ассоциативный массив). Другими словами, доступ к значению каждого столбца можно получить как по числовому индексу, так и по названию столбца. Предположим, результат запроса хранится в массиве с именем `$row`, тогда доступ к его элементам можно получить следующим образом:

```
$row[1]
$row[4]
```

[1] Массив представляет собой набор данных, объединенных под одним именем.

```
$row["name"]
$row["price"]
```

Здесь `[1],[4]`- числовые индексы столбцов `name` и `price` (нумерация столбцов начинается с нуля), `["name"]` и `["price"]`- названия столбцов.

Пример 10. Отображение всех строк таблицы `productFAM` (листинг 35).

Листинг 35. Извлечение всех строк из таблицы (файл `view_table.php`)

```
<HTML>
<BODY>
<h1>Отображение данных</h1>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
/* получение в переменную result результата SQL-запроса на выборку всех записей */
$result=mysql_query("SELECT * FROM productFAM");
/* проверка успешности выполнения SQL-запроса, если отрицательный результат – выход */
if(!$result) exit(mysql_error());
// открытие тега таблицы для вывода результата запроса
echo "<TABLE>";
// цикл для вывода строк результата запроса
while($row=mysql_fetch_array($result))
{
// открыть тег строки таблицы
echo "<TR>";
// открыть тег поля таблицы и вывести содержимое столбцов
// нумерация столбцов ведется с нуля!
echo "<TD>".$row[0]."</TD><TD>".$row[1].
"</TD><TD>".$row[2]."</TD><TD>".$row[3]."</TD>
<TD>".$row[4]."</TD>";
// закрыть тег строки таблицы
echo "</TR>";
}
// закрытие тега таблицы
```

```
echo "</TABLE>";
// закрытие соединения с сервером баз данных
mysql_close($connect);
?>
</BODY>
</HTML>
```

Примечание – Комментарии, которые в тексте листинга записывают после символов /* или //, можно не писать.

Пример 11. Отображение строк таблицы, удовлетворяющих определенному условию. Напишем листинг для извлечения из таблицы `productFAM` записей по поставкам яблок (листинг 36).

Листинг 36. Выборка записей из таблицы (файл `select_table.php`)

```
<HTML>
<BODY>
<h1> Выборка записей </h1>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
/* получение в переменную result результата SQL-запроса на выборку записей
по условию */
$result=mysql_query("SELECT * FROM productFAM
WHERE name LIKE 'яблоки'");
/* проверка успешности выполнения SQL-запроса, если не результат (отрица-
тельный результат) – выход */
if(!$result) exit(mysql_error());
// открытие тега таблицы для вывода результата запроса
echo "<TABLE>";
// цикл для вывода строк результата запроса
while($row=mysql_fetch_array($result))
{
// открыть тег строки таблицы
echo "<TR>";
// открыть тег поля таблицы и вывести содержимое столбцов
// нумерация столбцов ведется с нуля!
echo "<TD>".$row[0]."</TD><TD>".$row[1].
"</TD><TD>".$row[2]."</TD><TD>".$row[3]."</TD>
```

```
<TD>".$row[4]."</TD>";
// закрыть тег строки таблицы
echo "</TR>";
}
// закрытие тега таблицы
echo "</TABLE>";
// закрытие соединения с сервером баз данных
mysql_close($connect);
?>
</BODY>
</HTML>
```

Изменение содержимого записей таблицы

Для изменения данных требуется всего лишь написать соответствующий SQL-запрос. Например, требуется изменить цену яблок с 3 100 на 2 900 в таблице `productFAM`. Для начала следует установить соединение с базой данных аналогично тому, как это сделано в предыдущих разделах. Потом выполняется SQL-запрос, изменяющий заданную строку в таблице (листинг 37).

Листинг 37. Изменение строки в таблице

```
<?php
$query="UPDATE productFAM SET price=2900 WHERE
name='яблоки'";
$result=mysql_query($query) or die ("Ошибка за-
проса:".mysql_error());
?>
```

Удаление данных из таблицы

Для удаления данных из таблицы используется SQL-запрос `DELETE`. В запросе обычно присутствует условие `WHERE`, идентифицирующее те данные, которые следует удалить. Если по ошибке опустить условие (задается после ключевого слова `WHERE`), то будут удалены все записи из таблицы, так что в этом случае следует быть особенно внимательным. Команда удаления из таблицы `productFAM` записей по продукту «яблоки»:

```
DELETE FROM productFAM WHERE name='яблоки'";
```

Листинг 38 демонстрирует использование в PHP-сценарии команды удаления записей в таблице `productFAM`, в которых наименование продукта «яблоки».

Листинг 38. Удаление данных (файл `delete_table.php`)

```
<HTML>
<BODY>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
// создаем запрос на удаление строк
$query="DELETE FROM productFAM WHERE name='
яблоки'";
$result=mysql_query($query)
or die ("Ошибка запроса: ".mysql_error());
$query="SELECT * FROM productFAM";
$result=mysql_query($query)
or die ("Ошибка запроса: ".mysql_error());
echo "<TABLE>";
echo "<TR>";
echo "<TH>Наименование</TH><TH>Цена</TH>";
echo "</TR>";
while ($row=mysql_fetch_array($result))
{
echo "<TR>";
echo "<TD>".$row[1]."</TD><TD>".$row[4]."</TD>";
echo "</TR>";
}
echo "</TABLE>";
mysql_close ($connect);
?>
</BODY>
</HTML>
```

Функция определения количества строк в результирующем наборе

Одной из распространенных задач при выборке из базы данных является определение числа записей, которые возвращает функция `mysql_query()`. Для этого предназначена функция, имеющая следующий синтаксис:

```
mysql_num_rows(результат_запроса)
```

В качестве единственного аргумента `результат_запроса` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

Функция `mysql_num_fie1ds (результат_запроса)` позволяет определить число столбцов в результирующем наборе.

Лабораторная работа 4
РАБОТА С SQL-СЕРВЕРОМ СРЕДСТВАМИ PHP

Порядок выполнения работы

1. Проверьте наличие таблицы `productFAM` (`FAM` – ваша фамилия, набранная латинскими буквами) в базе данных `studentzaot`, созданной вами с помощью программы mysqladmin.

2. С помощью PHP-редактора (PHP Expert Editor) или блокнота напишите следующие скрипты:

2.1. Скрипт соединения с сервером MySQL и открытия базы данных (см. листинг 33, информацию по параметрам соединения уточните у преподавателя). Сохраните скрипт в файл с именем `config1.php` в свою рабочую папку на компьютере.

2.2. Скрипт на добавление записей в созданную таблицу (см. листинг 34). Сохраните скрипт в файл с именем `insert_table.php` в свою рабочую папку.

2.3. Скрипт на отображение всех записей, имеющихся в созданной таблице (см. листинг 35). Сохраните скрипт в файл с именем `view_table.php` в свою рабочую папку.

2.4. Скрипт на выборку записей из таблицы по определенному условию и отображению результата (см. листинг 36). Сохраните скрипт в файл с именем `select_table.php` в свою рабочую папку.

2.5. Скрипт на удаление записей из таблицы с заданным условием (см. листинг 38). Сохраните скрипт в файл с именем `delete_table.php` в свою рабочую папку.

Примечание – В каждом из скриптов должны обязательно присутствовать команды по подключению к серверу и по завершению соединения с сервером баз данных (их можно включить в скрипт через файл `config1.php` командой `include`).

3. Визуально проверьте правильность написания скриптов.

4. Скопируйте все написанные вами скрипты на учебный сервер университета в папку, указанную преподавателем.

5. Проверьте работу скриптов на учебном сервере. Для этого загрузите браузер Internet Explorer, и в адресную строку браузера запишите название PHP-сценария и место его нахождения на сервере.

Пример: если PHP-сценарий с именем `insert_table.php` скопирован на учебный сервер в папку `STUD_PHP_Z`, то в адресную строку браузера нужно набрать:

```
http://uchserv/STUD_PHP_Z/insert_table.php
```

В отчет по лабораторной работе предоставляются рабочая папка пользователя на сервере, названная по фамилии (фамилия набирается по-английски) с сохраненными текстами скриптов на PHP, и демонстрация работы скриптов на учебном сервере.

3.4. Использование HTML-форм для передачи параметров в скрипты PHP

Реализация передачи параметров методом POST

Протокол HTTP, лежащий в основе WWW, допускает передачу данных с помощью метода GET или POST. По умолчанию используется метод GET.

Передача данных методом GET не всегда является удобной по следующим причинам:

- пользователь может видеть значение параметров и легко подделывать их в строке запроса (GET-параметры передаются через HTTP-заголовки);
- объем передаваемой информации через GET-параметры ограничен (как правило, 8 кбайт).

Существует еще один способ передачи данных – передача через тело HTML-документа. Для этого предназначен метод POST. Чтобы передать данные из формы обработчику методом POST, атрибуту `method` тега `<form>` необходимо присвоить значение POST.

Рассмотрим применение метода POST на конкретном примере. Схема предлагаемого проекта передачи параметров из форм в сценарии PHP изображена на рисунке 25.

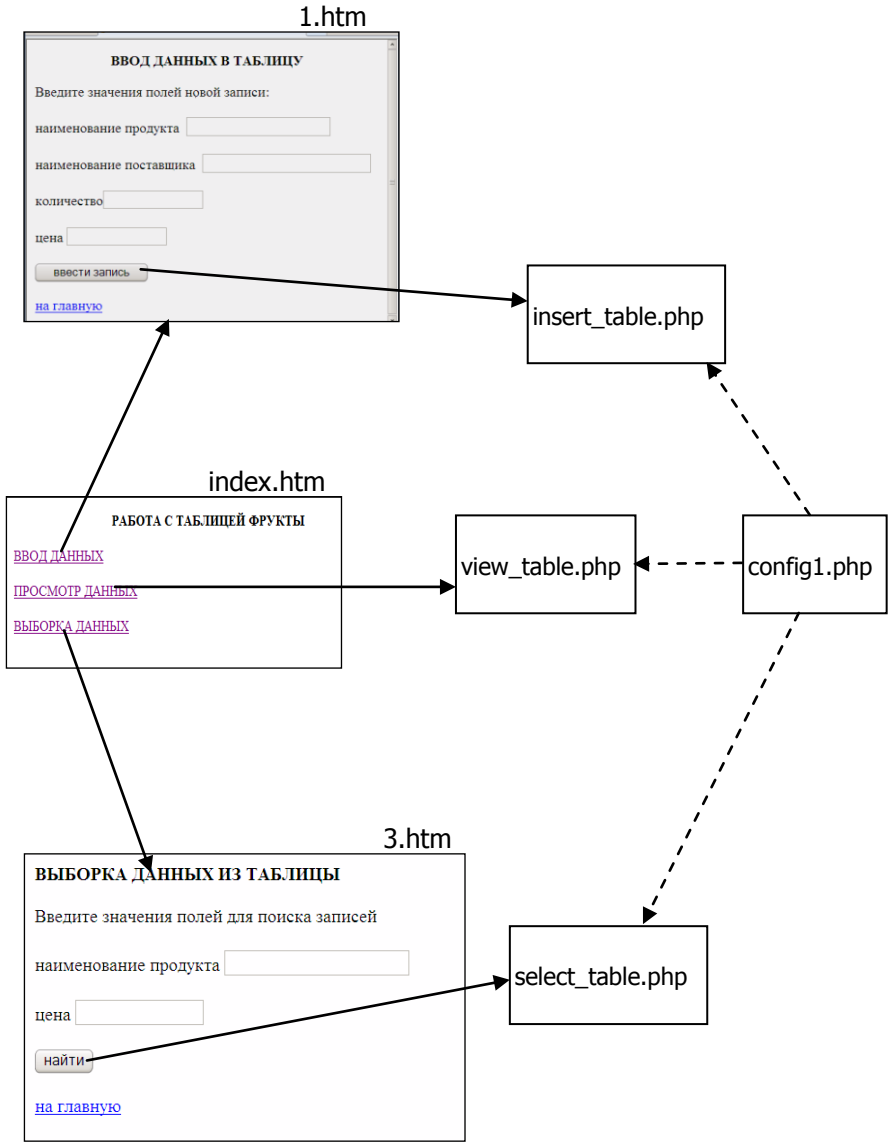

Рисунок 25 – **Схема взаимодействия HTML-форм с PHP-файлами**

Реализация передачи параметров при работе с таблицей `productFAM` представлена в примере 12.

Пример 12. Разработаем форму главного меню, которая будет содержать перечень основных режимов работы с таблицей, оформленных в виде гиперссылок (листинг 39 и рисунок 26).

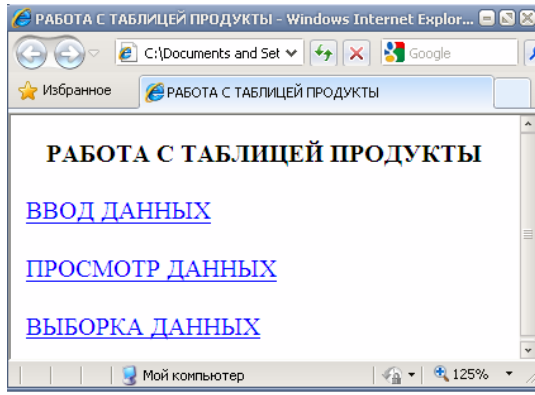

Рисунок 26 – **Окно главного меню**

Листинг 39. HTML-форма главного меню работы с таблицей (файл `index.htm`)

```
<html>
<head>
<title>РАБОТА С ТАБЛИЦЕЙ ПРОДУКТЫ</title>
</head>
<body>
<p align="center"><b>РАБОТА С ТАБЛИЦЕЙ ПРОДУКТЫ</b></p>
<!--гиперссылка открывает страницу с именем 1.htm с формой для заполнения полей таблицы-->
<p><a href="1.htm">ВВОД ДАННЫХ</a></p>
<!--гиперссылка открывает скрипт PHP, который выполняет просмотр записей таблицы-->
<p><a href="view_table.php">ПРОСМОТР ДАННЫХ</a></p>
<!--гиперссылка открывает страницу с именем 3.htm с формой для задания условий поиска записей таблицы-->
```

```
<p><a href="3.htm">ВЫБОРКА ДАННЫХ</a></p>
</body>
</html>
```

Примечание – Комментарии, которые в тексте листинга записывают между символами <!-- и -->, можно не писать.

Для организации ввода данных в таблицу разработаем форму, в которой у пользователя будут запрашиваться данные для заполнения таблицы. Назовем ее `1.htm`. Данные, введенные пользователем в форму, будут передаваться методом POST в скрипт `insert table.php`, который, в свою очередь, реализует SQL-запрос по вводу значений в таблицу (листинг 40 и рисунок 27).

Листинг 40. Форма для пункта меню «Ввод данных» (`1.htm`)

```
<html>
<body>
<p align="center> ВВОД ДАННЫХ В ТАБЛИЦУ </p>
<!-- тег открытия формы с передачей введенных данных методом POST в файл
insert_table.php -->
<form method="POST" action="insert_table.php">
<p>Введите значения полей новой записи:</p>
<!-- в поле name вводится наименование продукта-->
<p>наименование продукта  <input type="text"
name="name" size="25"></p>
<!--в поле post вводится наименование поставщика-->
<p>наименование поставщика  <input type="text"
name="postav" size="30"></p>
<!--в поле kol вводится количество продукта-->
<p>количество<input type="text" name="kol" size=
"16"></p>
<!--в поле price вводится цена продукта-->
<p>цена <input type="text" name="price" size=
"16"></p>
<!--задается кнопка для подтверждения ввода записи-->
<p><input type="submit" value="ввести запись"
name="OK"></p>
</form>
<p>
```

```
<!--гиперссылка для возврата на главную страницу-->
<p><a href="INDEX.htm">на главную</a></p>
</body>
</html>
```

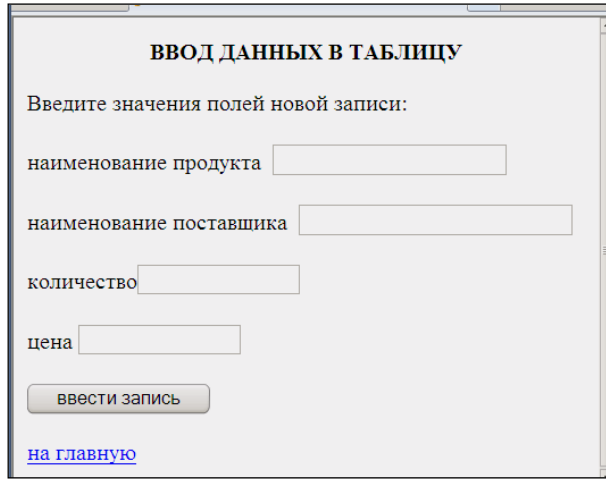

Рисунок 27 – **Окно первого пункта меню «Ввод данных»**

В связи с тем, что данные для заполнения таблицы передаются из HTML-формы `1.htm` в файл `insert_table.php` методом POST, внесем некоторые изменения в исходный файл `insert_table.php` (листинг 41).

Примечание – Файл `insert_table.php` описан в листинге 33.

Листинг 41. Откорректированный файл ***`insert_table.php`***

```
<HTML>
<BODY>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
/* в переменные сохраняются параметры, переданные методом POST из формы
1.htm */
$name=$_POST["name"];
$postav=$_POST["postav"];
```

```
$kol=$_POST["kol"];
$price=$_POST["price"];
// формирование SQL-запроса на добавление записей в таблицу
$query="INSERT INTO productFAM VALUES
(0,'$name','$postav','$kol','$price')";
// в переменную result заносится результат запроса
$result=mysql_query($query);
echo "Данные успешно добавлены";
// закрытие соединения
mysql_close($connect);
?>
</BODY>
</HTML>
```

Осталось разработать форму, в которую пользователь сможет ввести критерии выбора записей из таблицы. Назовем ее `3.htm` (третий пункт главного меню). Введенные данные методом POST будут передаваться в скрипт `select_table.php` (листинг 42, рисунок 28).

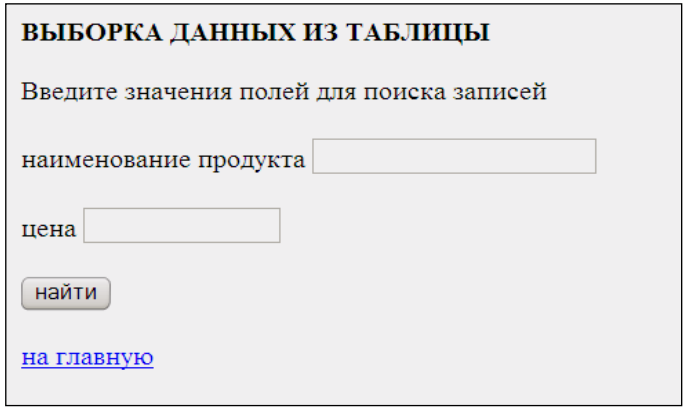

Рисунок 28 – **Окно формы пункта меню «Выборка данных»**

Листинг 42. Форма для третьего пункта меню «Ввод данных» (файл `3.htm`)

```
<HTML>
<BODY>
```

```
<p><b>ВЫБОРКА ДАННЫХ ИЗ ТАБЛИЦЫ</b></p>
<form method="POST" action="select_table.php">
<p>Введите значения полей для поиска записей:</p>
<p>наименование продукта <input type="text" name="name" size="25"></p>
<p>цена <input type="text" name="price" size="16"></p>
<p><input type="submit" value="найти" name="OK"></p>
</form>
<p><a href="INDEX.htm">на главную</a></p>
</BODY>
</HTML>
```

В связи с тем, что данные для заполнения таблицы передаются из HTML-формы `3.htm` в файл `select_table.php` методом POST, внесем некоторые изменения в исходный файл `select_table` (листинг 43).

Примечание – Файл `select_table.php` уже создан ранее на основе листинга 35.

Листинг 43. Откорректированный файл `select_table.php`

```
<HTML>
<BODY>
<?php
// подключение к серверу и открытие базы данных include "config1.php";
/* в переменные сохраняются параметры, переданные методом POST из формы 3.htm */
$name=$_POST["name"];
$price=$_POST["price"];
/* получение в переменную result результата SQL-запроса на выборку записей по условию */
$query="SELECT * FROM productFAM WHERE name LIKE '$name' and price LIKE '$price'";
// проверка успешности выполнения SQL-запроса
$result=mysql_query($query);
// если не результат (отрицательный результат) – выход
if(!$result) exit(mysql_error());
// открытие тега таблицы для вывода результата запроса
```

```
echo "<TABLE>";
// цикл для вывода строк результата запроса
while($row=mysql_fetch_array($result))
{
echo "<TR>";
// открыть тег строки таблицы
// открыть тег поля таблицы и вывести содержимое столбцов
// нумерация столбцов ведется с нуля!
echo "<TD>".$row[0]."</TD><TD>".$row[1].
"</TD><TD>".$row[2]."</TD><TD>".$row[3]."</TD>
<TD>".$row[4]."</TD>";
// закрыть тег строки таблицы
echo "</TR>";
}
// закрытие тега таблицы
echo "</TABLE>";
// закрытие соединения с сервером баз данных
mysql_close($connect);
?>
</BODY>
</HTML>
```

Лабораторная работа 5
ИСПОЛЬЗОВАНИЕ HTML-ФОРМ ДЛЯ ПЕРЕДАЧИ ПАРАМЕТРОВ В СКРИПТЫ PHP

Порядок выполнения работы

1. Напишите или проверьте наличие файла `config1.php`, в котором выполняется соединение с сервером MySQL и открытие базы данных (см. листинг 33).

2. Создайте HTML-форму главного меню (`index.htm`) работы с таблицей, а также форму (`1.htm`) для ввода значений полей в таблицу и форму (`3.htm`) для указания условий отбора полей из таблицы.

Формы можно создавать в редакторе FrontPage или в любом текстовом редакторе. При создании форм используйте листинги 39, 40, 42.

3. С помощью PHP-редактора (PHP Expert Editor) или текстового редактора Notepad (Блокнот) откорректируйте ранее написанные скрипты `insert_table.php`, `select_table.php`. Для корректировки воспользуйтесь листингами 41 и 43.

4. Самостоятельно создайте форму для удаления записей из таблицы по определенному условию. Откорректируйте ранее написанный скрипт `delete_table.php` (см. листинг 38). Добавьте пункт «Удаление данных» в главное меню приложения `index.htm`.

5. Визуально проверьте правильность скриптов.

6. Скопируйте все написанные вами файлы на учебный сервер университета в папку, указанную преподавателем.

Проверьте, после операции копирования папка на сервере должна содержать следующие файлы:

- `config1.php`;
- `index.htm`;
- `1.htm`;
- `3.htm`;
- `insert_table.php`;
- `view_table.php`;
- `select_table.php`;
- `delete_table.php`.

7. Проверьте работу скриптов на учебном сервере. Для этого загрузите браузер Internet Explorer и в адресную строку браузера запишите название главной страницы `index.htm` и место ее нахождения на сервере.

Примечание – Если главная страница `index.htm` скопирована на учебный сервер в папку `STUD_PHP_Z`, то в адресную строку браузера нужно набрать:

```
http://uch.btu/STUD_PHP_Z/index.htm
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Разработка веб-интерфейса работы с таблицами базы данных MySQL

В соответствии с вариантом задания, указанным ниже, создайте HTML-форму для отображения главного меню (основные режимы работы), а также HTML-формы для добавления, удаления, поиска и отображения записей таблицы базы данных. Напишите скрипты на PHP для обработки этих форм.

Результатом самостоятельной работы являются работающие страницы на сервере.

Вариант 1

Таблица с информацией о студентах и их распределении по местам практики: фамилия, год рождения, пол, группа, факультет, наименование предприятия (организации), город.

Вариант 2

Таблица с информацией об автомобилях: номер, год выпуска, марка, цвет, состояние, фамилия владельца, адрес.

Вариант 3

Таблица с информацией о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.

Вариант 4

Таблица с информацией о пропусках занятий студентами в течение семестра: фамилия, группа, факультет, дата, количество часов пропусков, причина пропусков (уважительная, неуважительная).

Вариант 5

Таблица с информацией о сотрудниках, имеющих компьютер: название отдела, фамилия сотрудника, данные о компьютере (тип процессора, объем оперативной памяти, установленная операционная система).

Вариант 6

Таблица с информацией о заказах, оформленных сотрудниками фирмы: фамилия сотрудника, оформившего сделку; наименование товара; сумма заказа; название фирмы-клиента; фамилия заказчика.

Вариант 7

Таблица с информацией об оценках, полученных студентами на экзаменах: фамилия, группа, наименование дисциплины, номер билета, оценка, преподаватель.

Вариант 8

Таблица с информацией о преподавателях кафедр: название кафедры, фамилия преподавателя, должность, ученая степень, читаемые курсы.

Вариант 9

Таблица с информацией об авторах веб-сайтов и их статьях: имя, учетная запись, пароль, тематика, заголовок статьи.

Вариант 10

Таблица с информацией о списке рассылки и подписчиках: тема и содержание письма, дата отправки, имена и адреса подписчиков.

Вариант 11

Таблица с информацией о специальностях, на которых обучаются студенты: имя, фамилия, название специальности, курс, группа.

Вариант 12

Таблица стола заказов ресторана: дата проведения мероприятия; фамилия, имя, отчество заказчика; общая сумма заказа; количество персон; вид торжества.

Вариант 13

Таблица с информацией о поставках продукции: дата, наименование продукции, единица измерения, количество, цена, номер накладной.

Вариант 14

Таблица учета валютных операций: дата, наименование валюты, курс, количество единиц, признак операции (покупка или продажа), общая стоимость.

Вариант 15

Таблица с информацией о таможенных операциях: дата, признак операции (ввоз или вывоз), группа товара, наименование товара, количество единиц, стоимость товара, сумма таможенной пошлины.

Вариант 16

Таблица с информацией о вкладах в банк: дата; фамилия, имя, отчество вкладчика; срок вклада; процентная ставка; сумма вклада.

Вариант 17

Таблица с информацией о реализации товаров: наименование товара, дата продажи, количество проданного товара, розничная цена, оптовая цена.

Вариант 18

Таблица учета командировок сотрудников: фамилия, имя, отчество сотрудника; должность; дата начала командировки; количество дней; расходы на транспорт; прочие расходы; аванс.

Вариант 19

Таблица учета дежурств сотрудников: фамилия, имя, отчество сотрудника; должность; подразделение (отдел); дата дежурства; время дежурства.

Вариант 20

Ведомость расчета заработной платы сотрудников: отдел; фамилия, имя, отчество сотрудника; должность; оклад; надбавки; премия.

Вариант 21

Таблица учета материальных ценностей: наименование товарно-материальной ценности, балансовая стоимость, количество (на балансе), фактическое количество, наименование отдела.

Вариант 22

Таблица учета оплаты коммунальных услуг: дата оплаты; фамилия, имя, отчество квартиросъемщика; наименование услуги; сумма оплаты; пеня.

Вариант 23

Таблица по учету выдачи кредитов юридическим лицам: дата оформления, номер договора, наименование юридического лица, адрес, сумма кредита, срок, годовая процентная ставка.

Вариант 24

Таблица штатного расписания организации: наименование отдела

или подразделения, наименование должности, количество штатных единиц, оклад.

Вариант 25

Таблица с информацией учебных планов университета: наименование изучаемой дисциплины, наименование кафедры, номер семестра, форма контроля знаний (экзамен или зачет), количество часов.

Вариант 26

Таблица с информацией туристских фирм: наименование турфирмы (туроператора), адрес, номер тура, страна, количество дней отдыха, стоимость, периодичность тура.

Вариант 27

Таблица с информацией о репертуаре кинотеатров: наименование кинотеатра, наименование фильма, период проката, признак (отечественный или зарубежный), длительность фильма, цена билета.

Вариант 28

Таблица с информацией о маршрутах поездов пригородного сообщения: наименование маршрута, номер поезда, периодичность маршрута, время отправления поезда, цена билета.

Вариант 29

Таблица с информацией об оформленных заказах в магазине «Компьютер-маркет»: дата оформления заказа; номер заказа; фамилия, имя, отчество заказчика; домашний адрес; стоимость покупки; номер счета-фактуры (перечень приобретенных комплектующих).

Вариант 30

Таблица с информацией учета больничных листов в поликлинике: номер больничного листа; дата начала больничного; дата окончания больничного; фамилия, имя, отчество врача; фамилия, имя, отчество больного; домашний адрес; место работы.

Вариант 31

Таблица с информацией учета услуг стоматолога: дата оказания услуги; фамилия, имя, отчество пациента; домашний адрес; вид услуги (стоматология или протезирование); наименование услуги; стоимость услуги.

Вариант 32

Таблица с информацией о продаже-покупке валюты в киосках: дата операции, тип операции (1 – покупка, 2 – продажа), наименование валюты, курс, количество, сумма продажи.

Вариант 33

Таблица расчета амортизации товарно-материальных ценностей (ТМЦ): наименование отдела, наименование ТМЦ, количество единиц ТМЦ (на балансе), фактическое количество, балансовая стоимость, годовая амортизация в процентах, сумма амортизации.

Вариант 34

Таблица с информацией о списании продовольственной продукции: номер акта списания, дата, наименование продукции, единица измерения, количество испорченной или просроченой продукции, цена.

Вариант 35

Таблица учета оплаты за стационарный телефон: дата оплаты; фамилия, имя, отчество плательщика; виды услуг (АПУС, МТР, абонентская плата); начислено за месяц; сумма оплаты; пеня.

Вариант 36

Таблица с информацией о прохождении тестов: фамилия тестируемого, группа, наименование теста (высшая математика, информатика, политология), количество правильных ответов, процент правильных ответов (от 0 до 100), преподаватель.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Дюбуа, П. MySQL. Полное и исчерпывающее руководство по применению и администрированию баз данных MySQL 4, а также программированию приложений : [пер. с англ.] / П. Дюбуа. – М. : Вильямс, 2004. – 1056 с.

Дюбуа, П. MySQL. Сборник рецептов : [пер. с англ.] / П. Дюбуа. – СПб. : Символ-Плюс, 2004. – 1056 с.

Хольцнер, С. PHP в примерах : [пер. с англ.] / С. Хольцнер. – М. : Бином-Пресс, 2007. – 352 с.

Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. А. Прохоренок. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2009. – 880 с.

Мотев, А. А. Уроки MySQL. Самоучитель / А. А. Мотев. – СПб. : БХВ-Петербург, 2006. – 208 с.

Документация по MySQL // База данных MySQL [Электронный ресурс]. – М., 2000. – Режим доступа : http://www.mysql.ru/docs/.

PHP manual // PHP : Hypertext Prepocessor [Electronic resource]. – US, 1997. – Mode of access : http://www.php.net/manual/ru/.

СОДЕРЖАНИЕ

Учебное издание

ИНФОРМАЦИОННЫЕ РЕСУРСЫ

ТЕХНОЛОГИИ РАБОТЫ С КЛИЕНТ-СЕРВЕРНЫМИ СУБД (НА ПРИМЕРЕ СУБД MySQL)

Практикум
к лабораторным занятиям для студентов заочной формы получения высшего образования специальности 1-26 03 01 «Управление информационными ресурсами»

Авторы-составители:
Заяц Татьяна Александровна
Заяц Вячеслав Михайлович

Редактор М. П. Герасенко
Технический редактор И. А. Козлова
Компьютерная верстка Н. Н. Короедова

Подписано в печать 10.07.13. Бумага типографская № 1.
Формат 60 × 84 $^{1}/_{16}$. Гарнитура Таймс. Ризография.
Усл. печ. л. 4,18. Уч.-изд. л. 4,00. Тираж 70 экз.
Заказ №

Учреждение образования
«Белорусский торгово-экономический университет потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.
ЛИ № 02330/0494302 от 04.03.2009 г.

Отпечатано в учреждении образования
«Белорусский торгово-экономический университет потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.

БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ
УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»

Кафедра информационно-вычислительных систем

ИНФОРМАЦИОННЫЕ РЕСУРСЫ

ТЕХНОЛОГИИ РАБОТЫ С КЛИЕНТ-СЕРВЕРНЫМИ СУБД (НА ПРИМЕРЕ СУБД MySQL)

Практикум
к лабораторным занятиям для студентов заочной формы получения высшего образования специальности 1-26 03 01 «Управление информационными ресурсами»

Гомель 2013